SABIN BURAGA
(coord.)

Aplicaţii Web la cheie

Seria *Web* este coordonată de Sabin Buraga
(Facultatea de Informatică, Universitatea „Al.I. Cuza", Iaşi)

Sabin Buraga, lector la Facultatea de Informatică a Universităţii „Al.I. Cuza" din Iaşi, este titularul cursurilor *Tehnologii Web*, *Reţele de calculatoare* şi *Proiectarea interfeţelor-utilizator*. Coordonează un grup de interes în domeniul tehnologiilor Web şi are ca direcţii de cercetare Web-ul semantic şi interacţiunile între sisteme distribuite pe scară largă. Mai multe detalii privind activităţile pe care le desfăşoară sunt disponibile pe Web la **http://www.infoiasi.ro/~busaco/**. De acelaşi autor, la Editura Polirom au apărut: *Proiectarea siturilor Web* (2002), *Programare Web în bash şi Perl* (colaborare cu V. Tarhon-Onu şi Şt. Tanasă, 2002), *Atelier de programare în reţele de calculatoare* (colaborare cu G. Ciobanu, 2001).

© 2003 by Editura POLIROM

www.polirom.ro

Editura POLIROM
Iaşi, B-dul Carol I nr. 4, P.O. BOX 266, 700506
Bucureşti, B-dul I.C. Brătianu nr. 6, et. 7, ap. 33, O.P. 37; P.O. BOX 1-728, 030174

Descrierea CIP a Bibliotecii Naţionale a României:

Aplicaţii Web la cheie: studii de caz implementate în PHP / coord.: Sabin Buraga – Iaşi: Polirom, 2003

272 p.; 24 cm

ISBN: 973-681-456-4

I. Buraga, Sabin (coord.)

004

Printed in ROMANIA

Sabin Buraga
(coord.)

Ciprian Ginghină, Bogdan Manolache, Cristian Nechita,
Cristina Oprea, Alexandrina Orzan, Constantin Puşcaşu,
Marius Ropotă, Iavi Rotberg, Silvana Solomon

Aplicaţii Web la cheie

Studii de caz implementate în PHP

POLIROM
2003

Tablă de materii

Prefaţă 9

CAPITOLUL 1 – **O bibliotecă pe Web** (Bogdan Manolache) 13

1. **Argument** 13
2. **Interfaţa** 13
3. **Baza de date** 18
4. **Partea de programare PHP** 23
   - 4.1. Crearea meniului 24
   - 4.2. Listarea cărţilor 28
   - 4.3. Modulul de căutare 36
   - 4.4. Topul celor mai accesate cărţi 40
5. **La sfârşit de capitol** 42
   - *Comentarii bibliografice* 43

CAPITOLUL 2 – **TopWeb – lupta pentru voturi** (Constantin Puşcaşu) 45

1. **Prezentare generală** 45
   - *Arhitectura generală a sistemului* 46
2. **Baza de date** 47
   - 2.1. Introducere 47
   - 2.2. Crearea tabelelor 48
3. **Managementul utilizatorilor** 53
   - 3.1. Introducere 53
   - 3.2. Înregistrarea utilizatorilor 53
   - 3.3. Autentificarea utilizatorilor 54
     - *Funcţia login($ID, $password)* 55
     - *Funcţia is_logged()* 57
     - *Funcţia logout()* 58
   - 3.4. Editarea datelor despre conturile utilizatorilor 59
4. **Contorizarea voturilor** 62
   - 4.1. Introducere 62
   - 4.2. Algoritmul de contorizare a voturilor 63
5. **Contorizarea vizitatorilor** 67
   - 5.1. Introducere 67
   - 5.2. Algoritmul de contorizare a vizitatorilor 67
6. **Concluzii** 69
   - *Comentarii bibliografice* 70

CAPITOLUL 3 – *E-Shop IT* – dezvoltarea unui magazin virtual (Ciprian Ginghină) 71
1. Obiectivele şi cerinţele unui magazin virtual 72
Ce trebuie să ştim despre un e-shop? 72
Care sunt principalele obiective ale unui e-shop? 73
Care sunt principalele cerinţe ale unui e-shop? 73
2. Soluţia software 74
3. Arhitectura aplicaţiei 74
3.1. Clasificarea în module şi interacţiunea dintre ele 78
3.2. Autentificarea utilizatorului 82
3.3. Afişarea dinamică a categoriilor 82
3.4. Afişarea dinamică a produselor 87
3.5. Coşul de cumpărături 94
3.6. Serviciul de plată prin transfer bancar 95
3.7. Căutarea produselor după anumite criterii 97
3.8. Administrarea aplicaţiei 97
Adăugarea înregistrărilor 99
Modificarea înregistrărilor 101
Ştergerea înregistrărilor 101
Concluzii 101
4. Manual de utilizare 107
5. La final... 107
Resurse bibliografice 107

CAPITOLUL 4 – *E-Pizza* – de la browser la WAP
(Cristian Nechita, Marius Ropotă, Iavi Rotberg) 109
1. Descriere generală 109
1.1. Obiective 110
1.2. Cerinţe 111
1.3. Analiza aplicaţiei 111
Modulul pentru efectuarea de comenzi on-line 112
Modulul de administrare a sitului Web 113
Modulul de generare a graficelor 113
Modulul de rezervări on-line 114
Modulul de plată electronică 115
Modulul de comandare prin WAP 115
1.4. Arhitectura software a aplicaţiei 115
Modulul pentru efectuarea de comenzi on-line 119
Modulul de administrare a sitului 122
Modulul de generare a graficelor 125
Modulul de rezervări on-line 130
Modulul de plată electronică 132
Modelul de comandare prin WAP 136
2. Ghid de utilizare 136
Prezentarea aplicaţiei pentru clienţi 137
Utilizarea coşului de cumpărături 138
Prezentarea aplicaţiei pentru administrator 143
3. Concluzii 144
Comentarii bibliografice 145

CAPITOLUL 5 – Geometrie... virtuală (cu VRML, SVG şi PHP) (Alexandrina Orzan) 145
1. Web 2D şi Web 3D 145
1.1. Problema „concretă" şi soluţia ei 145
Formularea problemei 145
Soluţie 146
1.2. Realitate virtuală (VR) şi VRML 146
Ce înseamnă realitate virtuală? 146
Ce este VRML? 147
Cum pot fi vizualizate lumile VRML? 147
Ce este necesar pentru a crea lumi VRML ? 148
1.3. Grafica vectorială şi SVG 148
Introducere în grafica vectorială 148
Ce este SVG ? 148
Caracteristici ale limbajului 149
2. GraphX3D – arhitectura conceptuală 150
2.1. Design-ul sitului 150
Organizarea informaţiei 151
Structura sitului 152
2.2. Design-ul interfeţei 154
Pagini de sine stătătoare 154
Definirea navigării în sit 155
Design-ul graficii 155
2.3. Design-ul paginii 156
Ierarhie vizuală 156
Consecvenţă 156
Dimensiunea paginii 157
3. GraphX3D – implementare 157
3.1. Organizarea fişierelor din sit 157
3.2. Scrierea codului 158
SVG 159
VRML 161
XML 165
PHP 166
4. Manual de utilizare 167
Cerinţe software 171
Cerinţe hardware 171
Comentarii bibliografice 172

CAPITOLUL 6 – Dynamic Charts – grafice în PHP (Cristina Oprea, Sabin Buraga) 173
1. Preambul 173
2. Biblioteca GD 174
2.1. Caracterizare şi scurt istoric 174
Scurt istoric 174
Domenii de utilizare 175
Restricţii în utilizarea bibliotecii GD 175
2.1. Funcţii PHP pentru utilizarea bibliotecii GD 176
Funcţii de creare, distrugere, încărcare şi salvare a imaginilor 176
Funcţii de desenare 178
Funcţii de manipulare a culorilor 179

Funcţii pentru utilizarea fonturilor ... 180
Alte funcţii utile ... 180
3. Prezentarea aplicaţiei *Dynamic Charts* ... 182
3.1. Utilizarea aplicaţiei ... 183
Paşi de urmat ... 183
3.2. Detalii de implementare ... 186
Utilizarea funcţiilor bibliotecii GD ... 187
4. În loc de concluzii ... 200
Comentarii bibliografice ... 200
CAPITOLUL 7 – Accesul la baze de date prin servicii Web (Silvana Solomon) ... 201
1. O introducere în servicii Web ... 201
1.1. Standarde ... 202
1.2. RPC şi XML-RPC ... 203
Privire de ansamblu ... 203
XML-RPC vs. *SOAP* ... 204
2. XML-RPC pentru PHP ... 205
2.1. Clase XML-RPC ... 205
Crearea unui client cu `xmlrpc_client` ... 205
Crearea unui server cu `xmlrpc_server` ... 206
Signatura metodelor ... 207
Raportarea erorilor ... 208
2.2. Funcţionalitatea aplicaţiei ... 208
2.3. Biblioteca XML-RPC pentru PHP furnizată de *Incutio* ... 213
Cererea ... 213
Răspunsul ... 213
Erorile ... 213
Specificaţia pentru interoperabilitatea codului de eroare ... 214
Conversia automată a tipurilor ... 214
Extensii XML-RPC suportate ... 214
Construcţia clientului de bază ... 215
Depanare ... 215
Construcţia unui server de bază ... 216
2.4. Biblioteca PHP *ADOdb* pentru lucrul cu bazele de date ... 218
Un exemplu – comanda Select ... 219
3. Aplicaţia *PhoneBillings* ... 221
4. Concluzii ... 228
Comentarii bibliografice ... 229
CAPITOLUL 8 – Aplicaţii propuse (Sabin Buraga) ... 231
ANEXA A – Detalii de instalare (Bogdan Manolache, Sabin Buraga) ... 237
ANEXA B – Limbajul PHP pe scurt (Sabin Buraga) ... 241
ANEXA C – Procesarea documentelor XML în PHP (Sabin Buraga) ... 247
ANEXA D – Biblioteca *XML22* (Iavi Rotberg) ... 253
ANEXA E – Biblioteca *JpGraph* (Iavi Rotberg) ... 257
ANEXA F – Introducere în WML (Marius Ropotă, Sabin Buraga) ... 261
Bibliografie generală ... 267

Prefaţă

Programatorul este un creator de universuri
pentru care este în întregime responsabil.
Joseph Weizenbaum

Cartea de faţă – coordonată de *Sabin Buraga* şi scrisă de un colectiv de studenţi şi absolvenţi ai Facultăţii de Informatică a Universităţii „Alexandru Ioan Cuza" din Iaşi – îşi propune să prezinte şapte aplicaţii Web concepute în *PHP: Hypertext Processor* (unul dintre cele mai utilizate servere de aplicaţii Web în prezent – atât pe plan mondial, cât şi pe cel local –, funcţionând în conjuncţie cu *Apache*, cel mai popular server Web). Aceste aplicaţii pot fi considerate „complete" din punctul de vedere al etapelor de analiză, proiectare, implementare şi menţinere a unui produs software, pe care trebuie să le aibă în vedere orice specialist – arhitect, programator, designer, administrator etc. – în domeniul tehnologiilor Web.

Această apariţie poate fi considerată un eveniment în peisajul publicistic informatic actual deoarece, în primul rând, constituie o prezentare a unor contribuţii originale realizate de (foarte) tineri – viitori sau deja actuali – programatori Web. În al doilea rând, materialul a fost redactat cu intenţia de a putea fi parcurs şi înţeles de orice persoană interesată de realizarea, cu efort minim, a aplicaţiilor Web. Astfel, s-a urmărit familiarizarea cititorilor începători cu modul de programare PHP şi lucrul cu baze de date pentru Web, dar şi safisfacerea programatorilor cu mai multă experienţă prin oferirea unor soluţii practice de implementare a unor diverse situri (din domenii precum *e-commerce* sau educaţional), soluţii înglobând diferite tehnologii Web actuale, precum familia de limbaje XML şi limbajele înrudite (*SVG – Scalable Vector Graphics*, *WML – Wireless Markup Language*, *XML-RPC* – baza pentru servicii Web, *X3D* – utilizat la modelarea lumilor virtuale). Actualii şi viitorii programatori Web au posibilitatea punerii în practică şi experimentării diferitelor „reţete" şi soluţii la cheie pentru a atinge deziderateul unei aplicaţii de anvergură, precum autentificarea şi administrarea utilizatorilor, trimiterea unor diverse tipuri de conţinut către navigatorul vizitatorilor, managementul sesiunilor, implementarea coşului de cumpărături, asigurarea ergonomiei sitului etc. Din acest punct de vedere, cartea se doreşte a fi o continuare firească – la nivel aplicativ – a problematicilor detaliate în volumul *Proiectarea siturilor Web* scris de Sabin Buraga şi apărut la Editura Polirom în anul 2002. Elevii din clasele mari şi studenţii de la facultăţile cu profil informatic (şi nu numai) vor afla direcţiile de urmat în vederea realizării de aplicaţii Web care să aspire la – şi chiar să atingă – statutul de aplicaţii profesioniste. De asemenea, cartea se adresează tuturor programatorilor Web şi specialiştilor în domenii precum bazele de date, interacţiunea om-calculator sau grafica digitală.

Al treilea motiv, poate mai puţin evident, al apariţiei unei astfel de cărţi este acela de a oferi dezvoltatorilor Web în formare, aflaţi la început de drum, diverse exemple de stiluri de gândire şi de realizare a unor tipuri de situri Web, prin intermediul unor studii de caz reprezentând soluţii ingenioase şi elegante, pe cât posibil eficiente. Arhitecţii Web cu experienţă vor putea ei înşişi imagina şi construi (universuri) alternative, posibil mai reuşite, la cele propuse de această lucrare.

Structura cărţii

Dezbătând o anumită aplicaţie, fiecare capitol cuprinde formularea cerinţelor pe care trebuie să le satisfacă aplicaţia, detalierea modului de implementare şi structurare a părţilor componente, ilustrarea celor mai interesante fragmente de cod-sursă şi prezentarea unui scurt manual de utilizare. La finalul capitolului se oferă un set cuprinzător de referinţe bibliografice suplimentare – tipărituri sau adrese Web – astfel încât cititorul să poată aprofunda materialul expus.

Primul capitol, conceput de *Bogdan Manolache*, explică modul de realizare a unei biblioteci electronice într-o asemenea manieră încât orice programator să poată dezvolta un sit de acest gen. Capitolul prezintă şi unele principii de design şi de programare Web, insistându-se asupra lucrului cu baze de date în PHP, folosindu-se unul dintre cele mai populare servere de baze de date – MySQL. Cititorii se vor putea iniţia în activităţile de editare a structurii şi conţinutului bazelor de date, capitolul prezentând o foarte utilă aplicaţie – *phpMyAdmin* – pentru aceasta. De asemenea, se expun unele dintre cele mai interesante aspecte ale sitului bibliotecii şi se oferă fragmentele de cod-sursă reprezentative ale implementării actuale. Acest capitol poate fi parcurs în conjuncţie cu anexa A (prezentând unele aspecte ale instalării sub Windows a serverului de aplicaţii PHP) şi anexa B (care face o succintă trecere în revistă a limbajului PHP).

Cel de-al doilea capitol, scris de *Constantin Puşcaşu*, este o continuare firească a subiectelor dezbătute de primul. Aplicaţia prezentată este un sit Web care gestionează topuri ale accesărilor de situri. Aici vor fi oferite detaliile necesare pentru crearea bazei de date, precum şi principalele tehnici folosite în procesul de contorizare a voturilor primite de pe siturile utilizatorilor. Se dau şi o serie de detalii privind modul de autentificare a vizitatorilor, folosind mecanismele puse la dispoziţie de PHP şi MySQL. Cititorilor interesaţi de managementul sesiunilor li se pun la dispoziţie unele amănunte sperăm utile.

Al treilea capitol prezintă o aplicaţie tipică de comerţ electronic: un magazin virtual. Autorul, *Ciprian Ginghină*, porneşte de la enumerarea principalelor obiective şi cerinţe ale unei astfel de aplicaţii şi continuă apoi cu detalierea unei posibile soluţii software – bazată pe Apache, PHP şi MySQL. Se dau explicaţii privind modulele aplicaţiei şi se oferă implementarea completă a uneia dintre cele mai importante componente: coşul de cumpărături. De asemenea, capitolul insistă asupra manierei de dezvoltare a serviciului de plată prin transfer bancar.

Subiectul principal al celui de-al patrulea capitol, avându-i ca autori pe *Cristian Nechita*, *Marius Ropotă* şi *Iavi Rotberg*, este înrudit cu cel al capitolului precedent. Este vorba tot de un sit de comerţ electronic, dar de această dată se descrie varianta Web a unei pizzerii. Cititorului i se oferă o soluţie completă pentru managementul informaţiilor referitoare la activitatea unei pizzerii, incluzând comandarea *on-line* de produse, rezervarea de mese în restaurant şi administrarea facilă prin intermediul unei interfeţe Web. De asemenea, se descrie şi o soluţie de accesare – via dispozitive WAP – a sitului, utilizându-se limbajul WML. Implementarea apelează la utilizarea limbajului JavaScript pentru realizarea operaţiei *drag-and-drop* la modulul de rezervare a meselor şi recurgere la două biblioteci PHP externe pentru procesarea documentelor XML şi generarea dinamică de grafice în cadrul modulului de administrare a aplicaţiei. Materialul acestui capitol este întregit de conţinutul anexelor D, E şi F.

Al cincilea capitol, redactat de *Alexandrina Orzan*, descrie paşii ce trebuie urmaţi pentru conceperea unui sit Web educaţional care construieşte reprezentarea grafică a unor curbe şi suprafeţe 3D. Se prezintă problematica abordată şi soluţia ingenioasă propusă, aplicaţia folosind două limbaje de redare a graficii Web 2D şi 3D – SVG şi X3D (reformulare în termenii XML a limbajului VRML). De altfel, merită menţionat faptul că formatul SVG este unul dintre formatele de viitor, standardizate de Consorţiul Web, destinate realizării de grafică vectorială pentru Web. Se urmăresc apoi etapele de creare a arhitecturii sitului şi modul de implementare a structurii obţinute, la aceasta contribuind mai multe „ingrediente": PHP, MySQL, SVG, VRML, XML şi JavaScript.

Următorul capitol – al şaselea – se situează tot în preajma graficii, însă de data aceasta *Cristina Oprea* şi *Sabin Buraga* prezintă modul în care se poate realiza în PHP, recurgând la funcţionalităţile bibliotecii GD, o aplicaţie Web de generare dinamică a unor grafice (*charts*) asemănătoare celor realizate de programele de calcul tabelar. Accesul la facilităţile oferite de biblioteca GD se poate face direct din limbajul PHP prin intermediul unor funcţii predefinite, utilizarea acestora fiind deosebit de comodă pentru programator. Implementarea aplicaţiei respectă paradigma programării orientate-obiect, bazându-se pe suportul obiectual oferit de PHP.

Capitolul al şaptelea, scris de *Silvana Solomon*, revizitează problematica accesării bazelor de date, dar în acest caz dintr-o perspectivă superioară, cea a serviciilor Web. Se prezintă principalele trăsături ale protocolului XML-RPC (o soluţie alternativă la mult mai complexul protocol SOAP), prin care este posibilă apelarea de proceduri la distanţă, mesajele schimbate fiind marcate în XML. Scopul acestui protocol este de a aşeza o fundaţie standardizată care să asigure compatibilitatea între medii eterogene, pentru implementarea de servicii Web bazate pe XML. Se exemplifică utilizarea tehnologiei XML-RPC pentru gestiunea unei baze de date Web indiferent de platformele folosite. Mai mult, capitolul ilustrează funcţionalităţile oferite de biblioteca *ADOdb* pentru lucrul cu bazele de date într-o manieră standardizată, orientată-obiect.

Ultimul capitol propune o serie de aplicaţii Web care pot fi implementate prin utilizarea cunoştinţelor acumulate în precedentele capitole ale cărţii.

Anexele intenţionează a fi îndrumare compacte, folositoare la înţelegerea şi/sau implementarea unora dintre subiectele puse la dispoziţie de volumul de faţă.

Materialul se încheie cu o bibliografie cuprinzătoare destinată lărgirii orizontului de cunoaştere a principalelor direcţii de dezvoltare a tehnologiilor Web actuale, în general, şi a programării folosind paradigma *open-source LAMP* (*Linux – Apache – MySQL – PHP*) utilizată cu precădere de fiecare dintre aplicaţiile prezentate, în special.

Mulţumiri

Demersul nostru n-ar fi putut ajunge la stadiul actual fără suportul şi satisfacţiile pe care coordonatorul acestei cărţi le-a avut de la studenţii săi, mai ales în timpul cursurilor şi laboratoarelor de *Tehnologii Web*, ca şi de la absolvenţii pe care i-a îndrumat în pregătirea lucrărilor de licenţă. Se cuvine menţionat faptul că majoritatea siturilor prezentate în volumul de faţă sunt părţi cuprinzătoare din proiectele de licenţă sau absolvire ale autorilor unora dintre capitole. Spaţiul tipografic nu a permis şi includerea altor aplicaţii de excepţie pe care unii dintre cei mai talentaţi studenţi – actuali absolvenţi – le-au implementat în ultimii ani. Le suntem tuturor recunoscători.

De asemenea, mulţumim colectivului grupului de interes *WebGroup* al Facultăţii de Informatică din Iaşi. Nu-i uităm nici pe prietenii şi colegii noştri; fără sugestiile, întrebările şi – mai ales! – criticile lor n-am fi putut progresa nicicând.

Coordonatorul cărţii şi al seriei *Web* din care face parte prezentul volum ţine, ca de obicei, să mulţumească familiei pentru înţelegerea şi încurajările exprimate de-a lungul timpului. Un gând special îi este destinat domnului *Dorel Lucanu* – îi mulţumim şi pe această cale pentru ideile de aplicaţii propuse în ultimul capitol şi, desigur, pentru discuţiile purtate pe marginea unor subiecte incluse în materialul de faţă.

Contact

Aşteptăm părerile şi reacţiile, sperăm pozitive, ale cititorilor la adresa **phpapps@infoiasi.ro**. Alte informaţii şi resurse privitoare la unele dintre aplicaţiile prezentate aici sunt disponibile pe situl Web dedicat cărţii la **http://www.infoiasi.ro/~phpapps/**. De asemenea, cititorii sunt încurajaţi să ia contact cu oricare dintre autorii capitolelor volumului.

Coordonatorul seriei *Web* poate fi contactat la **busaco@infoiasi.ro**.

Sabin Buraga
septembrie 2003, Iaşi

CAPITOLUL 1

O bibliotecă pe Web

Bogdan Manolache, `toni@MVDmedia.ro`

În acest capitol se explică modul de realizare a unei biblioteci electronice într-o asemenea manieră încât orice programator Web să poată dezvolta un sit de acest gen. Capitolul expune şi unele principii de design şi de programare Web, insistându-se asupra lucrului cu baze de date în PHP, folosindu-se serverul MySQL.

1. Argument

Primele biblioteci au apărut în urmă cu 4.000 de ani în centrele urbane din Mesopotamia şi erau folosite pentru a păstra informaţii despre contracte, taxe şi chitanţe. În prezent, cea mai mare bibliotecă din lume este Biblioteca Congresului din Statele Unite, având peste 120 milioane de înregistrări dintre care mai mult de 18 milioane sunt cărţi.

În viitor, orice instituţie de învăţământ va avea propria bibliotecă virtuală. O iniţiativă care vine să susţină această idee este decizia luată la MIT (Massachusetts Institute of Technology) ca de la începutul anului universitar 2002/2003 să fie disponibile *on-line* toate cursurile, gratuit, sub forma unei biblioteci electronice (virtuale).

Prerogativele bibliotecilor electronice sunt evidente dacă ne gândim că principalul dezavantaj al cărţilor este preţul. Bibliotecile disponibile pe Web oferă acces la informaţii gratuit, rapid şi de la orice calculator conectat la Internet. Misiunea principală a unei biblioteci este să colecteze, să organizeze, să păstreze şi să ofere acces cât mai uşor la cunoştinţe şi informaţii, funcţii oferite cu succes de către o bibliotecă virtuală, aşa cum va fi ilustrat şi de către aplicaţia prezentată în continuare.

Precizăm că tot ceea ce este expus în acest capitol reprezintă o particularizare specifică aplicaţiei prezentate, informaţiile parcurse de cititor folosind apoi – cu succes, sperăm – în crearea oricărei aplicaţii care utilizează PHP şi MySQL.

2. Interfaţa

Steven Johnson afirma despre designul interfeţei că reprezintă fuziunea dintre artă şi tehnologie („*The fusion of art and technology that we call interface design*"). Credem că

acest citat este foarte elocvent mai ales pentru interfeţele paginilor Web dinamice. Un sit Web dinamic este compus din două părţi fundamentale: *interfaţa* şi partea ascunsă utilizatorului final, cea care nu se vede, adică tocmai *funcţionalitatea* (partea de programare).

Interfaţa unei pagini Web reprezintă întruchiparea artei despre care vorbea Johnson, este ceea ce vede utilizatorul, este partea cu care utilizatorul interacţionează şi – poate cel mai important – reprezintă imaginea, identitatea unei pagini Web. Realizarea interfeţei unei pagini Web este foarte importantă, deoarece aceasta atrage şi încântă ochiul celui care interacţionează cu o pagină Web, ajutându-l să îi descopere funcţionalităţile. În general, interfaţa determină (sau nu) utilizatorii să revină la pagina respectivă. Din acest motiv interfaţa trebuie „să arate" foarte bine, să fie bine structurată şi, desigur, să fie ergonomică.

Interfaţa trebuie să comunice şi trebuie să comunice eficient. Mesajul pe care o interfaţă urmăreşte să-l transmită trebuie să se potrivească cu interpretarea utilizatorului. Cu alte cuvinte, utilizatorul trebuie să reuşească să interpreteze cu succes mesajul pe care autorul paginii Web doreşte să-l transmită prin felul în care a organizat textul, imaginile, animaţiile etc.

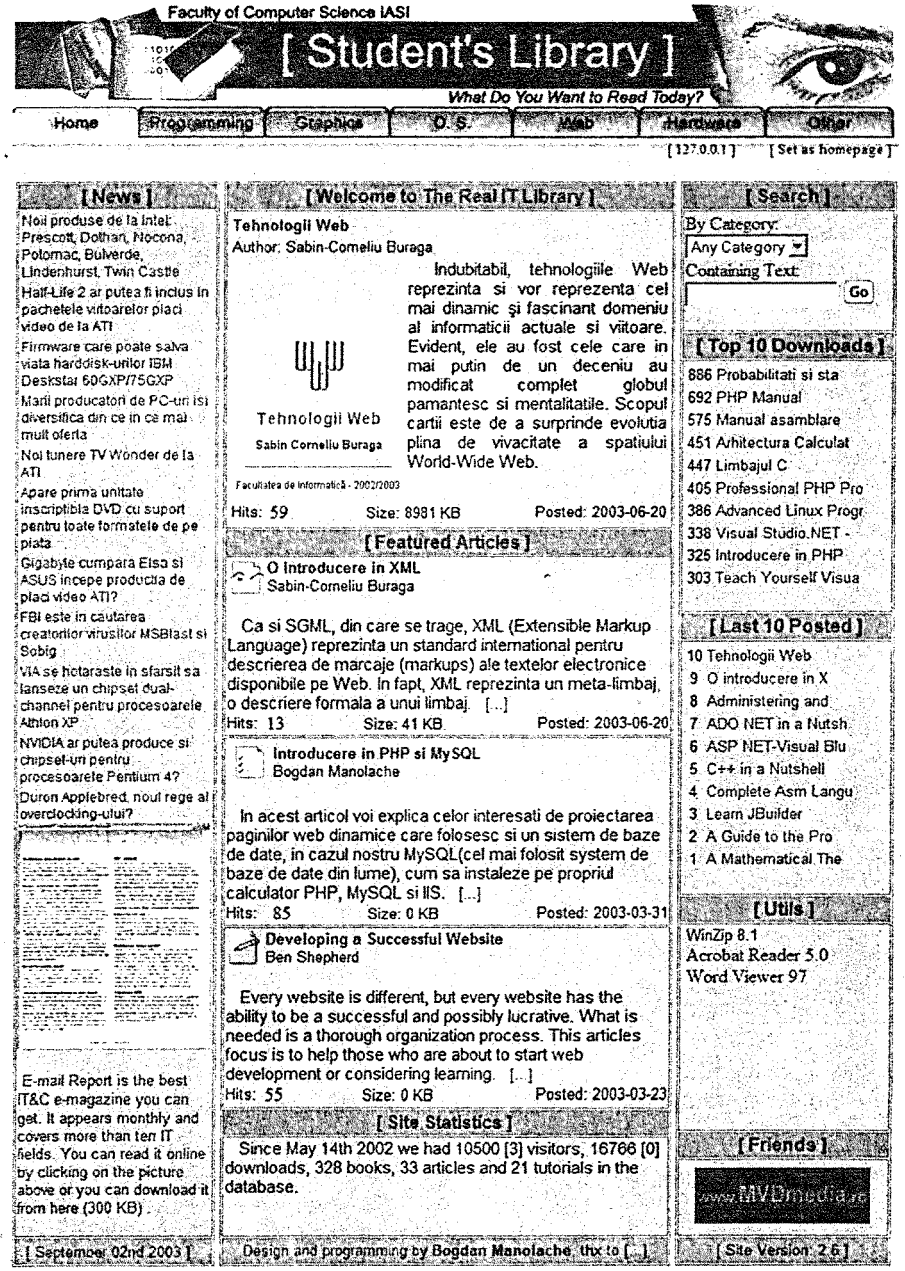
Acest capitol va pune accent mai mult pe partea de implementare a bibliotecii virtuale decât pe partea de design a interfeţei. Dar pentru că orice aplicaţie Web are nevoie de o interfaţă vom prezenta, pe scurt, metodele de realizare ale acesteia.

Pentru „fabricarea" unei interfeţe simple, pot fi folosite utilitare de editare HTML precum *Microsoft FrontPage* sau *Macromedia Dreamweaver* (soluţii comerciale) ori *1st Page* sau *HTML-Kit* (soluţii disponibile în regim *freeware*). Pentru realizarea elementelor grafice ale interfeţei sunt recomandate programe precum *Adobe Photoshop* sau *GIMP* (*GNU Image Manipulation Program*).

În imaginea alăturată se poate „studia" interfaţa actuală a bibliotecii virtuale. La adresa **www.MVDmedia.ro/lib.html** poate fi experimentat modul în care funcţionează biblioteca *on-line*. După cum se poate observa, interfaţa are aspectul unui portal – asta am şi urmărit în final, să realizăm un portal educaţional şi nu doar o bibliotecă electronică. Să vedem care sunt principalele elemente ale acestei interfeţe, ajunsă actualmente la „venerabila" versiune 2.6.

Primul element al interfeţei, comun majorităţii siturilor, este bannerul. Acesta cuprinde titlul paginii (*Student's Library*), un *motto* („*What Do You Want to Read Today?*") şi câteva elemente grafice: cărţile care prezintă specificul sitului şi un ochi care urmăreşte să inspire setea pentru cunoaştere – *Reading is the basis of knowledge*.

Următorul element – fiind de asemenea parte componentă a celor mai multe situri – este meniul. Acesta are rolul de a permite navigarea prin paginile sitului. În cazul nostru avem un meniu orientat orizontal care cuprinde o legătură către prima pagină a sitului (*Home*) precum şi categoriile tematice principale în care sunt împărţite cărţile: *Programming*, *Graphics*, *O.S.* (*Operating Systems*), *Web*, *Hardware* şi *Other*. Imediat sub butoanele meniului, unde vor apărea subcategoriile, este prezentă o legătură care permite setarea paginii drept implicită (*Set as homepage*) şi adresa de unde a fost accesat situl – pe care dacă o accesaţi va deschide pagina cu statisticile sitului (*Date, Time, IP, Host, Browser, Hits, Book*). Bara care apare sub meniu permite



Interfaţa bibliotecii virtuale (prima pagină – *index.php*)

o eventuală extindere viitoare a sitului prin adăugarea de noi legături. Atunci când proiectaţi un sit este bine să aveţi în vedere şi eventualele opţiuni care ar putea apărea în viitor şi să lăsaţi spaţiu pentru adăugarea acestora.

Acest sit nu a fost proiectat doar pentru a găzdui o bibliotecă virtuală, ci pentru a fi un portal educaţional pentru comunitatea pe care o deserveşte. Astfel, în coloana din partea stângă, puteţi observa ultimele ştiri din domeniul tehnologiei informaţiei (*News*), care sunt preluate dinamic de pe situl **www.div.ro**. Tot în partea stângă, în partea inferioară a paginii este prezentată o revistă electronică (*E-Magazine*) care poate fi citită *on-line* sau poate fi copiată pentru o lectură ulterioară. Revista apare lunar, cuprinzând ştiri şi articole pe diferite teme din informatică. Cititorii se pot abona şi ei la această revistă gratuită accesând situl **www.epress.ro**.

În partea centrală a paginii sunt prezentate cartea şi articolele cele mai interesante ale săptămânii (disponibile în format electronic, desigur). Sunt însoţite de informaţii suplimentare, cum ar fi numărul de descărcări, dimensiunea şi data publicării. Imediat sub ultimul articol sunt prezentate diverse statistici ale sitului: numărul total de vizitatori şi numărul de vizitatori din ultima zi, numărul total de descărcări şi numărul de descărcări din ultima zi, numărul de cărţi, numărul de articole şi numărul de tutoriale prezente în bibliotecă.

În partea dreaptă se găseşte modulul de căutare a cărţilor, foarte important pentru un astfel de sit. Căutarea poate fi efectuată după titlu, după categorie sau după subcategorie. Mai jos se află topul celor mai descărcate cărţi şi titlurile ultimelor cărţi adăugate în bibliotecă. Nu lipsesc nici utilitarele necesare parcurgerii cărţilor: *WinZip* (varianta *shareware*) pentru dezarhivare, *Acrobat Reader* pentru citirea documentelor de tip PDF şi *Word Viewer* pentru citirea documentelor de tip Word.

Situl este generat cu ajutorul a două fişiere PHP: `index.php` şi `programming.php`. Prima pagină, prezentată mai sus, a fost creată dinamic de codul `index.php`. Cea de-a doua pagină – generată de `programming.php` – poate fi privită în figura de mai jos. Această pagină are un conţinut dinamic care se modifică în funcţie de categoria şi subcategoria selectate. De exemplu, în imagine am selectat categoria *O.S.* şi apoi subcategoria *Linux*.

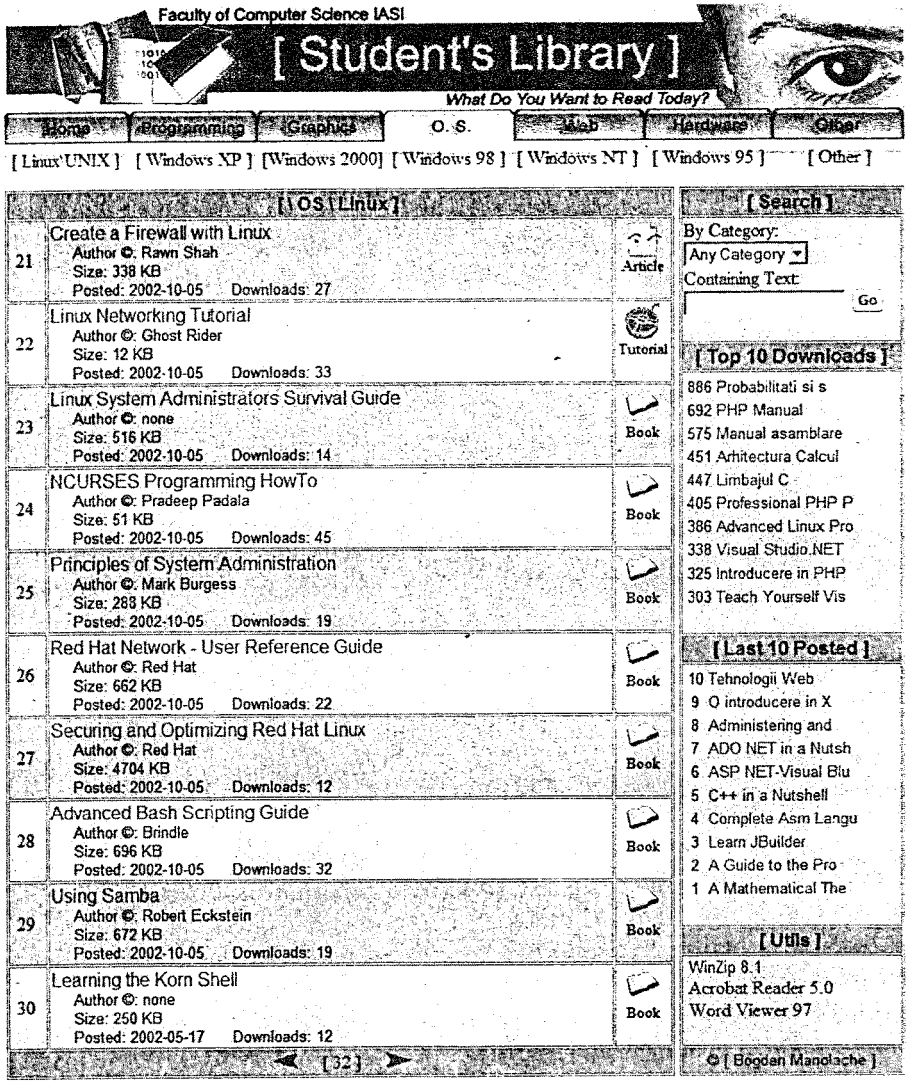
Ca modificări faţă de prima pagină observăm că bara de sub meniu conţine acum subcategoriile care aparţin categoriei *O.S.*: *Linux\UNIX*, *Windows XP*, *Windows 2000*, *Windows 98*, *Windows NT*, *Windows 95* şi *Other*. Biblioteca, în momentul de faţă, conţine şase categorii principale, iar fiecare categorie include până la şapte subcategorii.

Vom prezenta în continuare celelalte cinci categorii împreună cu subcategoriile aferente:

- Prima categorie este *Programming* şi are subcategoriile *Assembler*, *C\C++\VC++*, *C#*, *Java*, *Delphi*, *Visual Basic*, precum şi subcategoria *Other* care apare la fiecare categorie. După cum se poate observa, această categorie are ca temă programarea prezentând cărţi despre cele mai populare limbaje de programare.
- Cea de-a doua categorie se numeşte *Graphics*, are ca temă grafica digitală şi conţine subcategoriile *Photoshop*, *3D Studio Max*, *Corel Draw*, *Flash*, *Illustrator* şi *Other*.
- A treia categorie, *O.S.*, a fost deja prezentă mai sus.
- Urmează cea de-a patra categorie, numită *Web*, care conţine subcategoriile *HTML*, *XML*, *Web Design*, *ASP*, *PHP*, *Network* şi desigur *Other*. Această categorie

conţine documentaţii referitoare la programarea Web, proiectarea siturilor Web sau reţelele de calculatoare.

- Penultima categorie se numeşte *Hardware* şi conţine doar patru subcategorii: *Using PC*, *Modem*, *Upgrade* şi *Other*.
- A şasea categorie poartă numele de *Other* şi are subcategoriile *Database*, *Courses* şi *Other*.



Pagina dinamică a bibliotecii (*programming.php*)

Acesta este aşadar meniul sitului, care facilitează navigarea şi regăsirea documentaţiilor necesare pe baza împărţirii pe categorii şi subcategorii.

Pe această pagină observăm că sunt doar două coloane, cea din dreapta fiind similară coloanei de pe prima pagină. În coloana din stânga sunt enumerate cărţile din categoria şi subcategoria pe care ne aflăm. Fiecare carte are un număr de ordine, pe o pagină putând fi afişate maxim zece cărţi. În partea inferioară a paginii apar două săgeţi care permit afişarea următoarelor zece cărţi sau a celor zece precedente. Fiecare documentaţie are o pictogramă (*icon*) specifică, în funcţie de tipul acesteia: *Book*, *Article* sau *Tutorial*.

Înainte de a se trece la partea de programare din dezvoltarea unui sit, este recomandat ca interfaţa să fie terminată pentru că, după ce se începe implementarea propriu-zisă (de exemplu, în PHP), va fi relativ dificil să se modifice aspectul sitului.

Cam atât despre interfaţa bibliotecii virtuale. Nu uitaţi însă că interfaţa este poate cea mai importantă pentru utilizatori, uneori mult mai importantă decât conţinutul unui sit („*We find that people quickly evaluate a site by visual design alone*"* – Stanford Guidelines for Web Credibility).

3. Baza de date

Majoritatea siturilor care au conţinut dinamic folosesc un sistem de baze de date. Noi vom folosi MySQL, un sistem foarte rapid şi robust de management al bazelor de date (a se vedea şi cele discutate în capitolul secund al cărţii de faţă). Un sistem de baze de date permite stocarea, căutarea şi sortarea eficientă a datelor. MySQL foloseşte limbajul *SQL* (*Structured Query Language*), limbajul standard de interogare a bazelor de date.

Presupunem că aveţi instalat MySQL pe calculatorul pe care lucraţi (pentru o platformă Windows, serverul MySQL poate fi instalat automat de către aplicaţia *PHPTriad*, aceasta includzând, de asemenea, serverul Apache şi serverul de aplicaţii PHP – a se vedea şi CD-ul aferent cărţii S. Buraga, *Proiectarea siturilor Web*; pentru Linux, serverul MySQL e prezent în orice distribuţie actuală şi poate fi instalat direct, o dată cu instalarea sistemului de operare). A se consulta şi anexa A.

Vom prezenta în continuare două modalităţi de lucru cu sistemul de baze de date: una bazată pe comenzi, printr-o interfaţă de tip text, şi una utilizând o interfaţă grafică (disponibilă pe Web). Noi vom lucra cu baza de date folosind o aplicaţie cu interfaţă grafică.

Pentru a folosi interfaţa de tip text, disponibilă implicit în urma instalării MySQL, trebuie să rulăm clientul `mysql` care se găseşte, de obicei, în directorul `/usr/local/mysql/bin` pe Linux şi în directorul `c:\mysql\bin\` pe Windows (dacă aţi instalat *PHPTriad*, atunci serverul MySQL e localizat în directorul `c:\apache\mysql\bin\`). Pentru a vă conecta la baza de date, utilizaţi comanda `mysql -h <hostname> -u <username> -p`, unde înlocuiţi `<hostname>` cu adresa serverului pe care se află

* „Am descoperit că oamenii evaluează cu rapiditate un sit doar prin intermediul designului vizual".

instalat MySQL. Dacă este instalat pe calculatorul local atunci puteţi să folosiţi comanda fără opţiunea `-h <hostname>`. Parametrul `<username>` trebuie înlocuit cu numele de utilizator. Parametrul `-p` semnifică faptul că se va cere parola imediat după ce aţi rulat comanda. Va apărea un prompter de genul:

```
mysql>
```

Vom prezenta în continuare câteva comenzi utile în lucrul cu mediul de comandă text. Fiecare comandă MySQL trebuie să se termine cu punct şi virgulă.

Pentru a părăsi mediul MySQL folosim comanda `quit`.

Comanda `show databases;` afişează lista bazelor de date:

```
+------------+
| Database   |
+------------+
| mysql      |
| test       |
+------------+
2 rows in set (0.01 sec)
```

Pentru a şterge complet o bază de date folosim comanda `drop database <nume>;`.

Vom crea o bază de date, pe care o vom numi **lib**, folosind comanda `create database lib;`. Pentru a lucra cu această bază de date trebuie să o selectăm folosind comanda `use lib;`. Baza de date are în componenţă tabele, iar fiecare tabel este organizat în linii şi coloane, fiecare linie reprezentând o înregistrare şi fiecare coloană corespunzând unui anumit câmp din tabel. Fiecare tabel va conţine un anumit tip de informaţii. Noi vom crea tabelele folosind interfaţa Web, dar vom prezenta şi comenzile pentru interfaţa text. Pentru a crea o tabelă folosim:

```
mysql> create table <nume tabela> (
  -> <numele coloanei 1> <tipul col. 1> <detaliile col. 1>,
  -> <numele coloanei 2> <tipul col. 2> <detaliile col. 2>,
  -> ...
  -> );
```

De exemplu, pentru a crea o tabelă cu numele **books**, care să conţină câmpurile *bookid*, *title*, *cat* şi *subcat*, recurgem la următoarea comandă:

```
mysql> create table books (
  ->  bookid int not null auto_increment primary key,
  ->  title text,
  ->  cat int,
  ->  subcat int,
  -> );
```

Coloana **bookid** va conţine date de tip întreg, diferite de zero, se va mări cu unu la fiecare nouă înregistrare adăugată şi va fi cheie primară, adică va identifica în mod unic înregistrările din tabelă. Dacă dorim să vedem lista tabelelor din baza de date selectată folosim comanda `show tables;`, iar pentru a vedea informaţii despre

coloanele unei tabele utilizăm comanda `describe <nume tabela>;`. Pentru a şterge o tabelă folosim `drop table <nume tabela>`.

Dacă dorim să adăugăm date în tabelă utilizăm următoarea comandă, cu ambele variante de sintaxă:

```
mysql> insert into <nume tabela> set
   ->  numele_col1 = valoare1,
   ->  numele_col2 = valoare2,
   ->  ...
   -> ;
mysql> insert into <nume tabela>
   -> (numele_col1, numele_col2, ...)
   -> values (valoare1, valoare2, ...);
```

Pentru a vedea informaţiile introduse într-un tabel utilizăm comanda `select * from <nume tabela>;`. De exemplu, pentru a selecta doar informaţiile din coloanele **bookid** şi **title** folosim comanda `select bookid, title from books;`.

Dacă doriţi să modificaţi informaţiile introduse deja într-un tabel puteţi utiliza comanda `update` cu următoarea sintaxă:

```
mysql> update <nume tabela> set
   -> <nume_col>=<noua_valoare>, ...
   -> where <clauza where>;
```

iar pentru a şterge o înregistrare din tabel folosiţi:

```
mysql> delete from <nume tabela> where <clauza where>;
```

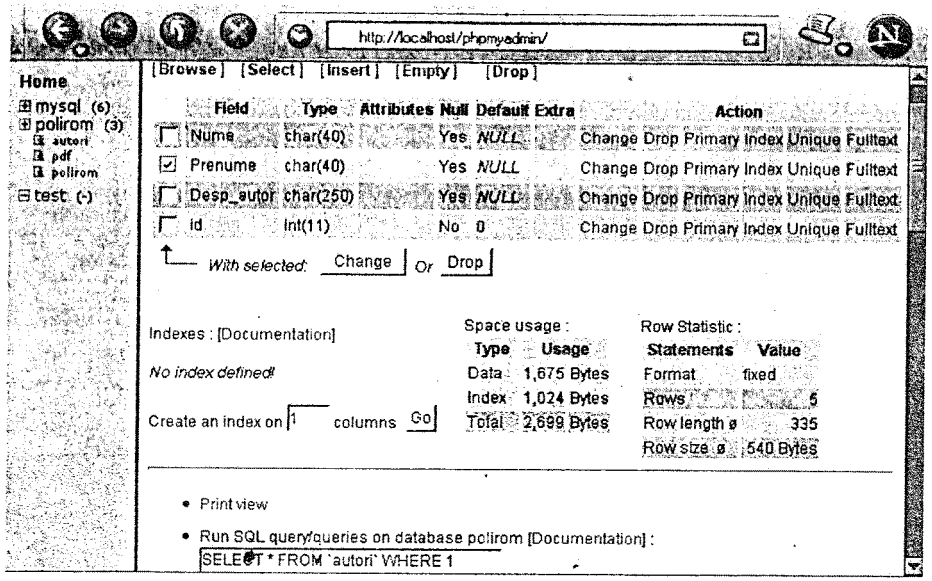
Acestea sunt cele mai uzuale comenzi MySQL. Pentru informaţii mai detaliate, folosiţi manualul – disponibil în diverse formate – pe care îl puteţi descărca de la adresa **www.mysql.com**.

În continuare, vom prezenta o aplicaţie PHP care va uşura mult lucrul cu bazele de date prin intermediul unei interfeţe vizuale. Există mai multe aplicaţii de acest gen, dar preferăm *phpMyAdmin* care poate fi descărcată gratuit de la adresa **www.phpmyadmin.net** (acest program este inclus şi în pachetul *PHPTriad*). Aplicaţia se prezintă sub forma unei pagini Web, ceea ce îi oferă avantajul unei instalări şi utilizări facile. Pentru a o accesa, în mod uzual trebuie să lansaţi un navigator Web şi ca adresă să furnizaţi `http://localhost/phpmyadmin/` (desigur, înainte de toate trebuie să fie pornite serverele Apache şi MySQL – vezi anexa A).

Interfaţa aplicaţiei *phpMyAdmin* este foarte simplu de folosit. Este împărţită în două coloane: coloana din stânga prezintă bazele de date create şi tabelele aferente, iar în coloana din dreapta apar acţiunile pe care le puteţi efectua în cazul fiecărei baze de date.

Să realizăm structura bazei de date. După ce aţi rulat *phpMyAdmin*, în coloana din dreapta puteţi crea o bază de date, să o numim **lib**. După ce aţi creat baza de date veţi observa că aceasta apare în coloana din stânga, iar în coloana din dreapta puteţi să creaţi tabele în această bază de date.

În continuare vom prezenta lista de tabele care trebuie create, împreună cu proprietăţile fiecărui câmp.



Interfaţa Web a aplicaţiei *phpMyAdmin*

Prima tabelă se numeşte **books**, are 9 câmpuri şi o vom folosi pentru a memora informaţiile referitoare la cărţi. Introduceţi deci în coloana din dreapta numele tabelei şi numărul de câmpuri, iar apoi apăsaţi **Go**. Va apărea o nouă pagină care conţine un tabel cu 9 linii (câte o linie pentru fiecare câmp al tabelei definite), iar fiecare linie are mai multe proprietăţi (coloane). Iată în continuare proprietăţile pe care le vom folosi

Field	Type	Default	Extra	Primary	Unique
bookid	int		auto_increment	x	x
title	text				
cat	int	0			
subcat	int	0			
author	text				
date	date				
link	text				
public	int	0			
type	text				

Câmpurile tabelei cu numele *books*

cel mai des: numele câmpului (*Field*), tipul câmpului (*Type*) şi valoarea implicită (*Default*). Acestea sunt informaţiile pe care trebuie să le introduceţi în tabel.

În acelaşi mod expunem informaţiile despre toate tabelele de care vom avea nevoie. Pentru a adăuga o nouă tabelă selectaţi baza de date din coloana din stânga şi procedaţi ca mai sus.

Field	Type	Default	Extra	Primary	Unique
catid	int		auto_increment	X	x
name	text				

Câmpurile tabelei *categories*

Tabela **categories** conţine categoriile în care sunt împărţite cărţile (cele 6 din meniul principal): Programming, Graphics, OS, Web, Hardware şi Other. După ce aţi creat tabela adăugaţi aceste categorii folosind legătura **Insert**. După ce aţi repetat de 6 ori pasul de adăugare puteţi vizualiza înregistrările folosind legătura **Browse**, iar din noua pagină puteţi edita sau şterge orice înregistrare.

Urmează tabela **download**, care memorează pentru fiecare carte câte descărcări s-au făcut.

Field	Type	Default	Extra	Primary	Unique
bokid	int	0		X	x
accesed	int	0		X	x

Câmpurile tabelei *download*

Tabela **subcat** conţine subcategoriile fiecărei categorii. Iată structura ei:

Field	Type	Default	Extra	Primary	Unique
scatid	int	0		X	x
subcatid	int	0		X	x
sname	text				

Câmpurile tabelei cu numele *subcat*

În tabelul următor sunt indicii unici ai categoriilor, indicii şi numele subcategoriilor, informaţii care vor trebui introduse în tabela **subcat**.

Numărul din prima coloană (*scatid*) ne va spune din ce categorie face parte fiecare subcategorie, iar numărul din cea de-a doua coloană (*subcatid*) ne va spune indicele subcategoriei.

scatid	subcatid	sname
1	1	Assembler
1	2	C
1	3	C Sharp
1	4	Java
1	5	Delphi
1	6	Visual Basic
1	7	Other
2	1	Photoshop
2	3	Corel Draw
2	4	Flash
2	5	Illustrator
2	7	Other
3	1	Linux
3	2	Windows XP
3	3	Windows 2000
3	4	Windows 98

scatid	subcatid	sname
3	5	Windows NT
3	6	Windows 95
3	7	Other
4	1	HTML
4	2	XML
4	3	WebDesign
4	4	ASP
4	5	PHP
4	6	Network
4	7	Other
5	1	Using PC
5	3	Upgrade
5	7	Other
6	1	Database
6	2	Courses
6	7	Other

Numele subcategoriilor din tabela *subcat*

Aceasta este structura bazei de date precum şi informaţiile care trebuie introduse în baza de date. Sperăm că v-aţi descurcat, fie prin intermediul interfeţei text care vine cu MySQL, fie prin intermediul interfeţei oferite de *phpMyAdmin*, fie cu ajutorul altui manager de baze de date pe care îl preferaţi.

4. Partea de programare PHP

Să vedem o scurtă prezentare a limbajului pe care îl vom folosi. *PHP* (*PHP: Hypertext Processor*) este un limbaj de tip script proiectat special pentru Web, care rulează pe server. În cadrul codului HTML al unei pagini Web putem insera cod PHP care va fi executat de fiecare dată când acea pagină este vizitată. Codul PHP este interpretat de către serverul Web, fiind generat cod HTML sau orice alt tip de conţinut (de exemplu fişiere grafice – a se vedea capitolul 6), care va fi văzut de vizitatorul paginii.

Limbajul PHP a fost conceput în anul 1994, iniţial fiind rezultatul muncii lui Rasmus Lerdorf. A fost adoptat apoi şi de alţi programatori talentaţi, trecând prin trei mari îmbunătăţiri până a ajuns la produsul pe care îl folosim astăzi, adică *PHP 4*. Actualmente, se lucrează la versiunea 5 a limbajului.

În ianuarie 2001 limbajul PHP era folosit pe aproape cinci milioane de situri Web din lume, iar acest număr a crescut rapid. În momentul editării acestui capitol

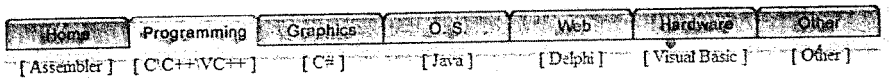
numărul domeniilor Internet care utilizează PHP este de aproximativ 13.500.000 (pentru o statistică la zi, vizitaţi adresa **http://www.php.net/usage.php**).

Orice secvenţă de cod PHP începe cu `<?php` sau `<?` şi se termină cu `?>`. Tot ceea ce se află cuprins între aceşti delimitatori este procesat de către interpretorul PHP, iar rezultatul este trimis *browser*-ului sub formă de cod HTML (sau orice altceva), navigatorul urmând a-l interpreta şi afişa pentru utilizator.

Pentru o trecere în revistă a limbajului PHP recomandăm parcurgerea anexei B.

4.1. Crearea meniului

Primul element de care ne vom ocupa este meniul bibliotecii. Acest meniu va trebui să apară pe cele două pagini ale bibliotecii virtuale: `index.php` (prima pagină) şi `programming.php` (pagina dinamică). Atunci când vom selecta o categorie va trebui să afişăm pe bara de sub meniu subcategoriile şi să listăm cărţile din prima subcategorie. La selectarea oricărei subcategorii vor fi listate cărţile care aparţin acelei subcategorii. Meniul bibliotecii virtuale, prezentat în figura următoare, este de fapt un tabel cu şapte coloane şi două linii. În fiecare celulă de pe rândul superior am inserat câte un buton grafic cu numele unei categorii. Această linie superioară care prezintă categoriile este fixă, fiind aceeaşi pe fiecare pagină. Se modifică doar butonul selectat, pentru a indica categoria pe care ne aflăm. Linia inferioară, care conţine denumirile subcategoriilor specifice fiecărei categorii, are un caracter dinamic, adică se schimbă în funcţie de categoria selectată, fiecare categorie având alte subcategorii. Celulele de pe linia a doua au pe fundal o imagine care imită o bară şi nu mai conţin butoane grafice, ci doar legături.



Meniul bibliotecii

Vom avea nevoie de numele categoriilor şi al subcategoriilor din baza de date. De fiecare dată când avem nevoie de orice informaţie stocată în baza de date trebuie să fim conectaţi la aceasta. În continuare vom prezenta secvenţa de cod pentru conectarea la baza de date, tot ceea ce se află între `/*` şi `*/` reprezentând comentarii explicative, nefiind interpretate:

```
<?php
    /* ne conectam la baza de date de pe calculatorul
       local cu numele de utilizator "dbuser"
       si cu parola dbpassword */
    $cnx = @mysql_connect ("localhost", "dbuser",
                           "dbpassword");


    if (!$cnx)
```



```
    {
      echo ("<p>Nu s-a putut realiza o conexiune la baza
      de date.</p>");
      exit();
    }
    /* selectam baza de date "lib" */
    if (! @mysql_select_db("lib", $cnx))
    {
      echo
        ("<p>Nu am putut localiza baza de date lib.</p>");
      exit();
    }
?>
```

Nu uitaţi să înlocuiţi adresa serverului pe care se află baza de date (*localhost*), numele de utilizator (*dbuser*) şi parola (*dbpassword*) cu cele specifice serverului vostru de baze de date. Această secvenţă de cod trebuie să apară în orice pagină în care vreţi să interacţionaţi cu baza de date. Apare o problemă: atunci când se va modifica numele bazei de date (*lib*), numele utilizatorului (*dbuser*) sau parola (*dbpassword*) va trebui să facem aceste modificări în fiecare fişier în parte. Acest neajuns poate fi rezolvat cu ajutorul funcţiei `include()`, care inserează, acolo unde este folosită, conţinutul fişierului pe care-l primeşte ca parametru. Vom salva secvenţa de cod de mai sus într-un fişier, să-l numim **dbconnect.php**, iar la începutul fiecărui fişier în care folosim baza de date scriem următoarea secvenţă de cod:

```
<?php
    include("../dbconnect.php");
?>
```

Acolo unde vom folosi această secvenţă de cod, linia `include(dbconnect.php)` va fi înlocuită cu fragmentul de cod dat mai sus, în felul acesta realizându-se conectarea la baza de date. În cazul nostru funcţia va apărea la începutul fişierelor `index.php` şi `programming.php`.

Fişierul **dbconnect.php** conţine informaţii importante care ar putea permite oricui să se conecteze la baza de date. În cazul în care interpretorul PHP nu mai funcţionează, fişierele PHP nu vor mai fi interpretate, ci va fi afişat codul-sursă al acestora. Pentru a evita acest lucru este recomandat ca fişierele critice să fie mutate cu un nivel mai sus decât directorul unde se află paginile Web, iar sintaxa funcţiei `include()` va fi `include("../dbconnect.php")`, sau chiar de forma `include("../../dbconnect.php")` dacă fişierul se află cu două niveluri mai sus în structura de directoare. De fiecare dată când aveţi o secvenţă de cod care se repetă în mai multe fişiere, puteţi folosi funcţia `include()` pentru a economisi spaţiu şi pentru a facilita modificările viitoare.

După ce aţi reuşit să creaţi meniul din punct de vedere vizual (încadrându-l într-un tabel şi folosind proprietăţi de stil corespunzătoare), vom adăuga împreună câte o legătură la fiecare buton. Să luăm, de exemplu, primul buton **Programming**. Acestuia

îi atribuim legătura spre adresa `programming.php?catid=1&subcatid=1`. În acest fel transmitem scriptului `programming.php` doi parametri (variabile): `catid` şi `subcatid`, amândoi având valoarea 1. Acesta este formatul de transmitere a parametrilor de la o pagină la alta prin metoda GET, putând transmite oricâţi parametri şi valorile acestora sub forma `pagina.php?parametru1=valoare1&parametru2=valoare2&....`

Valoarea 1 asociată variabilei `catid` ne va indica faptul că am apăsat butonul coresponzător categoriei 1 (*Programming*), şi vom selecta ca implicită subcategoria 1 a acestei categorii care, dacă vom consulta tabelul **subcat**, vom observa că se numeşte *Assembler*. În mod analog setăm legăturile pe următoarele butoane ale meniului:

Buton	Legătură
Programming	`programming.php?catid=1&subcatid=1`
Graphics	`programming.php?catid=2&subcatid=1`
O.S.	`programming.php?catid=3&subcatid=1`
Web	`programming.php?catid=4&subcatid=1`
Hardware	`programming.php?catid=5&subcatid=1`
Other	`programming.php?catid=6&subcatid=1`

Legăturile de pe butoane

Acest meniu va apărea pe ambele pagini (`index.php` şi `programming.php`) în mod identic. În schimb, pe pagina dinamică (`programming.php`) se află şi un submeniu, care este oarecum puţin mai complicat de realizat deoarece acesta se modifică în funcţie de categoria selectată. Iată codul care realizează submeniul primei categorii:

```
<td width="14%" align="center">
<font color="#335588">
<a href=
<?php echo("programming.php?catid=$catid&subcatid=1") ?> >

<?php if($catid==1) {echo("[ Assembler ]"); } ?>
<?php if($catid==2) {echo("[ Photoshop ]");} ?>
<?php if($catid==3) {echo("[ Linux\UNIX ]");} ?>
<?php if($catid==4) {echo("[ HTML ]");} ?>
<?php if($catid==5) {echo(" [ Using PC ] ");} ?>
<?php if($catid==6) {echo("[ Database ]");} ?>
</a>
</font>
</td>
```

Să analizăm această secvenţă de cod. Rezultatul codului este scris în interiorul primei celule de pe linia submeniului. Pentru restul celulelor codul este similar,

diferind doar denumirile subcategoriilor. Linia `<a href=<?php echo("programming.php?catid=$catid&subcatid=1") ?>>` reprezintă partea de început a definiţiei unei legături (reamintim că forma generală a definiţiei unei legături este `<a href="URL">Aceasta este o legatura</a>`). Mai sus, avem doar fragmentul `<a href="URL">`, iar în loc de `URL` am scris o linie de cod PHP: `<?php echo ("programming.php?catid=$catid&subcatid=1") ?>`. Funcţia predefinită `echo()` este folosită pentru afişare, în cazul nostru interpretarea funcţiei conducând la obţinerea definiţiei HTML pentru hiperlegături `<a href="programming.php?catid=1&subcatid=1">`, dacă este selectată categoria 1 (*Programming*). Reamintim, de asemenea, că un identificator prefixat de semnul „$" este o variabilă, iar variabilele în limbajul PHP nu trebuie să fie definite de un anumit tip, ele urmând a avea tipul valorii atribuite – a se vedea anexa B. Variabilele sunt înlocuite – atunci când sunt interpretate – cu valoarea lor, deci în cazul nostru atribuirea va fi `catid=1` deoarece am presupus că este selectată prima categorie, pentru care `$catid` are valoarea 1. Acest lucru înseamnă că atunci când vom selecta prima subcategorie a oricărei categorii se va deschide pagina corespunzătoare acelei categorii şi primei sale subcategorii. De exemplu, dacă ne-am fi aflat pe prima subcategorie a celei de-a treia categorii, am fi avut `catid=3` şi `subcatid=1`.

Următoarea linie, `<?php if($catid==1) { echo("[ Assembler ]"); } ?>`, testează dacă ne aflăm pe prima categorie, iar în caz afirmativ va afişa titlul primei subcategorii a acesteia, şi anume `[ Assembler ]`. Observăm că următoarea linie de cod, `<?php if($catid==2) {echo("[ Photoshop ]");} ?>`, verifică dacă este selectată a doua categorie (*Graphics*) şi în caz afirmativ afişează prima subcategorie corespunzătoare acestei categorii (*Photoshop*). Analog pentru celelalte linii de cod. Aceasta a fost secvenţa de cod pentru afişarea conţinutului primei celule a liniei subcategoriilor.

Vom prezenta în continuare codul pentru cea de-a doua celulă, corespunzător celei de-a doua subcategorii a fiecărei categorii:

```
<td width="15%" align="center">
<font color="#335588">
<a href=
<?php echo("programming.php?catid=$catid&subcatid=2") ?> >

<?php if($catid==1) {echo("[ C\C++\VC++ ]");}?>
<?php if($catid==2) {echo("[3D Studio Max]");}?>
<?php if($catid==3) {echo("[ Windows XP ]");}?>
<?php if($catid==4) {echo("[ XML ]");}?>
<?php if($catid==5) {echo("[ Modem ]");}?>
<?php if($catid==6) {echo("[ Courses ]");}?>
</a>
</font>
</td>
```

Următoarele celule vor conţine aceeaşi secvenţă de cod, cu singura diferenţă că vor apărea denumirile tuturor subcategoriilor cu indicele 3, 4, 5, 6, respectiv 7.

4.2. Listarea cărţilor

În continuare se prezintă secvenţa de cod (aparţinând fişierului `programming.php`) care realizează poate cea mai importantă funcţionalitate a bibliotecii: afişarea cărţilor.

4	Manual asamblare Author ©: none Size: 213 KB Posted: 2002-05-07 Downloads: 575	Book
5	Assembler Manual Author ©: none Size: 330 KB Posted: 2002-05-07 Downloads: 283	Book
6	Addendum - Intel Architecture Software Author ©: none Size: 126 KB Posted: 2002-05-07 Downloads: 56	Book

Listarea cărţilor din bibliotecă

În primul rând trebuie să adăugaţi nişte cărţi fictive în baza de date. Pentru aceasta deschideţi *phpMyAdmin*, selectaţi din coloana din dreapta baza de date **lib**, iar apoi selectaţi tabelul **books** care păstrează informaţiile despre cărţi. Pentru a introduce o nouă înregistrare (carte) în baza de date, amintim că puteţi folosi legătura **Insert** din coloana din stânga. Introduceţi informaţiile referitoare la carte, iar în câmpurile **cat** şi **subcat** treceţi indicele categoriei, respectiv al subcategoriei din care va face parte cartea. Repetaţi acest pas pentru fiecare carte în parte.

Vom lista cărţile într-un tabel la care vom adăuga o nouă linie pentru fiecare carte. Dar întâi trebuie să selectăm cărţile din baza de date. De exemplu, să se listeze numai resursele făcând parte din categoria şi subcategoria curente. Codul care selectează aceste cărţi din baza de date este:

```
$booklist = mysql_query
("select title,author,date,type,link,bookid,cat,subcat
from books where public>$public and cat=$catid and
subcat=$subcatid order by date desc limit $lim, $limm ");
```

Parametrul funcţiei `mysql_query()` este de fapt o interogare, o comandă MySQL, similară celor pe care le-am prezentat când am vorbit despre baze de date (a se revedea subcapitolul 3 al actualului capitol).

Interogarea SQL de mai sus selectează informaţii (`title, author, date, type, link, bookid, cat, subcat`) despre acele cărţi din tabelul `books` care respectă condiţia `where` şi anume: să fie publice (`public>$public`), să aparţină categoriei selectate (`cat=$catid`) şi respectiv subcategoriei selectate (`subcat=$subcatid`). Sunt de fapt trei condiţii legate între ele prin operatorul logic `and`.

Se poate observa faptul că fiecare înregistrare a unei cărţi din baza de date posedă şi un câmp numit `public`. Acesta poate avea valoarea 0, ceea ce înseamnă că acea carte nu este publică sau 1, însemnând că acea carte poate fi văzută de oricine. Informaţia din acest câmp va permite selectarea doar a cărţilor publice pentru un utilizator care nu este autentificat (logged) sau care provine de la o anumită adresă. De exemplu, dacă setaţi variabila `$public=-1`, atunci condiţia `public>$public`, unde `public=0` sau `public=1`, va fi mereu adevărată şi vor fi selectate toate cărţile, dar dacă `$public=0` vor fi selectate doar cărţile care sunt publice. În concluzie, dacă vrem ca un utilizator să poată accesa toate cărţile setăm variabila `$public=-1`, iar dacă dorim ca acesta să „vadă" doar anumite cărţi setăm `$public=0`. Şi să nu uităm de ultima parte a interogării, `order by date desc limit $lim, $limm`, care ordonează rezultatele descrescător după dată (cele mai recente cărţi vor fi listate primele), şi selectează doar 10 cărţi. În acest fel vom lista doar 10 cărţi pe fiecare pagină; dacă am lista toate cărţile pagina s-ar încărca mai greu, iar utilizatorul ar fi obligat să deruleze vertical conţinutul pentru a putea vedea toate cărţile (a se consulta şi capitolul 5 din *Proiectarea siturilor Web* de Sabin Buraga). Atunci când utilizatorul va dori să vadă lista următoarelor 10 cărţi selectate va folosi butonul de înaintare, în felul acesta vor fi listate următoarele cărţi. Analog şi pentru cele 10 cărţi precedente, cu singura diferenţă că vom scădea 10.

Codul care creează câte o nouă linie în tabel şi afişează informaţiile pentru fiecare carte este:

```
<?php
  $num = 1;
  // atata timp cat mai avem informatii de listat...
  while (($book = mysql_fetch_array($booklist))
          and ( $number < $limm))
  {
    // preluam in variabile locale informatiile
    // transmise de MySQL
    $title = $book["title"];
    $author = $book["author"];
    $date = $book["date"];
    $type = $book["type"];
    $link = $book["link"];
    $bookid = $book["bookid"];
    $cat = $book["cat"];
    $subcat = $book["subcat"];
    $ddown = mysql_query ("select accesed from download
                           where bokid=$bookid");
    $dwn = mysql_fetch_array($ddown);
    $accesed = $dwn["accesed"];

    // folosita pentru a afisa diferit randurile
    // din tabelul HTML
```

```
  $color = $num % 2;
  ?>
  <tr>
  <?php
  // incepem sa generam un rand
    if ($color == 0) { $color1 = "#DDDDDD" ;}
   else { $color1 = "#CCCCCC" ;}
  ?>
  <td bgcolor= "<?php echo($color1) ?>" width="30" align="center">
    <?php
      $number = $lim + $num;
      echo("$number");
    ?>
</td>
<td width="500" bgcolor="<?php echo($color1) ?>">
  <font face="Arial">
  <?php
   $path = mysql_query("select name, sname from categories,
   subcat where catid=$cat and subcatid=$subcat and scatid=$cat");
   $pathh = mysql_fetch_array($path);
   $pathcat = $pathh["name"];
   $pathscat = $pathh["sname"];
   // calea catre arhiva ZIP cu cartea
   $title1 = "$cale". "$pathcat" . "/" .
              "$pathscat/$title.zip";
   if($link == "") {
     ?>
     <a href = "<?php echo("$PHP_SELF?catid=$catid&subcatid=
     $subcatid&bokid=$bookid&lim=$lim&acces=$accesed&titl=
     $title1& select=$select&from=$from&where=$where&
     searchtext=$searchtext") ?> " > <?php
     echo("$title<br>") ?> </a>
     <?php
     }
     else {
       $link1="$cale"."$pathcat"."/"."$pathscat/$link"; ?>
       <a href = "<?php echo("$PHP_SELF?catid=
       $catid&subcatid=$subcatid&bokid=$bookid&lim=
       $lim&acces=$accesed&titl=$link1&select=$select&from=
       $from&where=$where&searchtext=$searchtext") ?>
       " style="text-decoration: none"> <?php
       echo("$title<br>") ?> </a>
     <?php
     }
```



```
    ?>
  <font size="2">
  <?php
    // informatii despre autor...
    $num++;
    if($author == "")
    {
      echo("Author &copy;: none");
    }
      else
    {
      echo("Author &copy;: $author");
    }
    // marimea arhivelor
    if($type == "a" and $link != "")
      {
        $siz = filesize("$link1");
      }
      else
      {
        $siz = filesize("$title1");
      }
    // alte informatii
    $siz = round( $siz / 1024 );
    echo("<br>Size: $siz KB");
    if((isset($select)) and ($select != ""))
      {
        echo("[ \\$pathcat \\$pathscat ]");
      }
    echo("<br>Posted: $date");
    echo("Downloads: $accesed");
 ?>
  </font></font>
</td>
<td align="center" valign="top" width="45"
    bgcolor="<?php echo($color1) ?>"><font size="2">
<?php
  // tipul de resursa
  if($type == "")
  { ?>
    <img border="0" src="resurse/book.gif"/><br />
    <?php echo("Book");
  }
  else if($type == "a")
```

```
        { ?>
          <img border="0" src="resurse/article.gif"/><br />
          <?php echo("Article");
        }
        else
        { ?>
          <img border="0" src="resurse/tutorial.gif"/><br />
          <?php echo("Tutorial"); } ?>
      </td>
      </font>
    </tr>
    <?php
      }
    ?>
```

Să analizăm în detaliu codul PHP dat mai sus.

```
while (($book = mysql_fetch_array($booklist))
        and ( $number < $limm))
```

Anterior am discutat despre funcţia `mysql_query()`. Aceasta returnează rezultatul interogării (primite ca parametru) sub forma unui tablou. În cazul nostru rezultatul interogării este în variabila de tip tablou `$booklist` (`$booklist = mysql_query(...)`), iar cu funcţia `mysql_fetch_array($booklist)` returnăm câte un element din acest tablou – corespunzător unei linii (înregistrări) din tabela interogată – la fiecare pas al buclei `while()`. În variabila `$number` memorăm numărul cărţii curente din cele care trebuie listate. Acest număr nu trebuie să depăşească numărul maxim de cărţi care trebuie listate până la această pagină (`$number < $limm`).

De fiecare dată când veţi dori să preluaţi informaţii din baza de date veţi avea nevoie de această pereche de funcţii: `mysql_query()` şi `mysql_fetch_array()`.

```
$title = $book["title"];
$author = $book["author"];
$date = $book["date"];
...
```

Cu aceste linii de cod obţinem în sfârşit informaţiile din baza de date. Variabila `$title` va conţine titlul cărţii, informaţie preluată din variabila de tip tablou `$book`. Celelalte variabile conţin informaţii despre carte, precum numele autorului sau data la care a fost publicată cartea în bibliotecă. Documentaţiile din bibliotecă au fost, generic, denumite *cărţi*, dar acestea pot fi tutoriale, articole sau cărţi (precum şi orice alt tip de resurse; de exemplu, fişiere audio). Acest lucru îl memorăm în câmpul **type** din baza de date care poate fi vid, deci implicit resursa reprezintă o carte, sau poate conţine litera **a**, ceea ce înseamnă că este articol, sau litera **t** dacă este vorba de un tutorial.


```
$ddown = mysql_query("select accesed from download where
bokid=$bookid");
$dwn = mysql_fetch_array($ddown);
$accesed = $dwn["accesed"];
```

Vă amintiţi că am creat în baza de date un tabel care se numeşte **download** şi care are două câmpuri: **bokid** şi **accesed**. Acest tabel va conţine, pentru fiecare carte, numărul de identificare şi de câte ori a fost descărcată. Secvenţa de cod de mai sus păstrează în variabila `$accesed` numărul de descărcări pentru cartea curentă.

```
<tr>
<?php
  if ($color == 0) { $color1 = "#DDDDDD" ;}
  else { $color1 = "#CCCCCC" ;}
?>
<td bgcolor= "<?php echo($color1) ?>" width="30" align="center">
  <?php
    $number = $lim + $num;
    echo("$number");
  ?>
</td>
```

Fiecare ciclu al buclei `while` realizează procesarea informaţiilor despre o carte şi afişarea acestor informaţii într-o nouă linie a tabelului (`<tr>`). Pentru a uşura diferenţierea cărţilor, fiecare linie a tabelului va avea o culoare diferită faţă de linia precedentă şi de linia anterioară. Vor exista două culori (nuanţe de gri): `#DDDDDD` şi `#CCCCCC`. Fiecare linie va fi împărţită în trei coloane, prima conţine numărul de ordine al documentaţiei (`echo("$number");`), a doua informaţiile despre documentaţie, iar a treia coloană va conţine tipul documentaţiei.

```
$path = mysql_query("select name,sname from categories,
subcat where catid=$cat and subcatid=$subcat and scatid=$cat");
$pathh = mysql_fetch_array($path);
$pathcat = $pathh["name"];
$pathscat = $pathh["sname"];
$title1 = "$cale" . "$pathcat" . "/" .
          "$pathscat/$title.zip";
```

Această secvenţă de cod urmăreşte să formeze legătura care va duce la descărcarea cărţii atunci când se va selecta titlul. Trebuie să precizăm că toate cărţile se află stocate pe *hard-disk* în directorul **bookdir**, fiecare într-un subdirector corespunzător categoriei şi subcategoriei din care face parte acea carte. Cărţile se află în aceste subdirectoare, împărţite astfel pe categorii şi subcategorii. În acest moment, pentru o carte ştim numărul de ordine al categoriei (`$cat`) şi numărul de ordine al subcategoriei (`$subcat`), iar în tabelul **categories** şi **subcat** fiecare denumire de categorie, respectiv subcategorie, are asociat un număr de ordine. Linia de cod `$path = mysql_query`

("select name, sname from categories, subcat where catid=$cat and subcatid=$subcat and scatid=$cat"); selectează denumirea categoriei şi a subcategoriei pentru cartea curentă.

Dorim să formăm calea de acces la carte din structura de directoare corespunzătoare acestei cărţi. Calea finală va fi stocată în variabila `$title1`, care este formată din numele directorului **bookdir** (la începutul fişierului *programming.php* definim variabila `$cale="bookdir/"`), al directorului care poartă numele categoriei (`$pathcat`), al directorului care poartă numele subcategoriei în care se află cartea (`$pathscat`) şi în cele din urmă din titlul cărţii la care adăugam extensia **.zip**. Fiecare carte este arhivată, numele fişierului fiind titlul cărţii.

```
      if($link == "") {
        ?>
        <a href = "<?php
echo("$PHP_SELF?catid=$catid&subcatid=$subcatid&bokid=
$bookid&lim=$lim&acces=$accesed&titl=$title1&select=
$select&from=$from&where=$where&searchtext=$searchtext") ?> " >
<?php echo("$title<br>") ?> </a>
        <?php
        }
        else {
          $link1="$cale"."$pathcat"."/"."$pathscat/$link";
?>
          <a href = "<?php
 echo("$PHP_SELF?catid=$catid&subcatid=$subcatid&bokid=$boo
kid&lim=$lim&acces=$accesed&titl=$link1&select=$select&from=$from
&where=$where&searchtext=$searchtext") ?> "
style="text-decoration: none"> <?php echo("$title<br>") ?>
</a>
        <?php
        }
        ?>
```

Unele documentaţii sunt de tip pagină Web şi nu are rost să le arhivăm, mai ales dacă vrem să le prezentăm pe prima pagină a bibliotecii. Alte documentaţii au denumiri conţinând caractere care nu pot apărea aşa cum sunt în adresele Web (de exemplu semnul „+" este înlocuit cu „%2B"). Pentru astfel de documentaţii există în baza de date un câmp denumit **link** care va conţine denumirea exactă, cu tot cu extensie şi având caracterele speciale din denumire înlocuite cu codul lor în hexazecimal. În secvenţa de cod anterioară verificăm dacă **link** este vid (`if($link == "")`), ceea ce înseamnă că această carte va avea extensia **.zip** şi nu conţine caractere speciale. În continuare vom forma legătura care duce la descărcarea cărţii. Variabila PHP predefinită `$PHP_SELF` conţine numele paginii Web curente la care adăugăm parametrii de care avem nevoie şi anume:

- `$catid` – numărul de identificare al categoriei;
- `$subcatid` – numărul de identificare al subcategoriei;

- `$lim` – numărul de cărţi pe care le-am listat până în acest moment;
- `$accesed` – numărul de descărcări pentru această carte;
- `$title1` – calea formată anterior de unde va fi descărcată cartea;
- `$from`, `$where` şi `$searchtext` – sunt folosite pentru modulul de căutare care va fi prezentat mai târziu.

Ramura `else` este pentru cazul în care câmpul **link** din baza de date conţine denumirea exactă a documentaţiei, iar singura diferenţă faţă de legătura anterioară este înlocuirea variabilei `$title1` cu variabila `$link1`, care va memora calea exactă către carte, formată din numele directorului **bookdir**, directorul care are denumirea categoriei, directorul care are denumirea subcategoriei din care face parte cartea şi numele exact al cărţii (`$link`).

Secvenţa de cod rămasă realizează afişarea celorlalte informaţii referitoare la documentaţia actuală: numele autorului, mărimea documentaţiei, data la care a fost publicată în bibliotecă, numărul de descărcări şi imaginea corespunzătoare tipului de documentaţie (tutorial, carte sau articol).

La începutul fişierului `programming.php` trebuie să verificăm dacă se doreşte descărcarea unei cărţi. Pentru aceasta, pur şi simplu verificăm dacă variabila `$acces` conţine vreo valoare sau nu. În caz afirmativ încărcăm calea către carte:

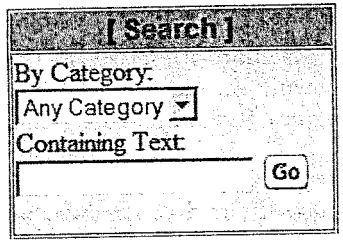
```
<?php
    $cale = "bookdir/";
    if ( isset($acces) )
      {
?>
<meta http-equiv="Refresh" content="0; url= <?php
echo($titl) ?> ">
```

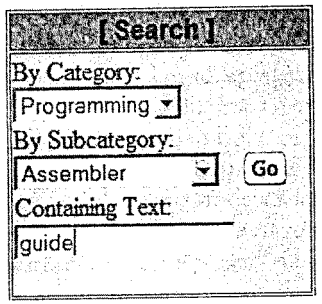
După ce a fost descărcată cartea, trebuie incrementat numărul de descărcări din baza de date pentru cartea respectivă. În primul rând trebuie să aflăm din baza de date valoarea veche a numărului de descărcări, incrementăm această valoare, iar apoi modificăm valoarea din baza de date cu noua valoare. Pentru a modifica o înregistrare din baza de date folosim comanda `update`, pe care am prezentat-o când am discutat despre baza de date.

```
if ( isset($acces) )
{
  $testacces = mysql_query("select accesed from download
where bokid=$bokid");
  $testaces = mysql_fetch_array($testacces);
  $accest = $testaces["accesed"];
  $accest++;
  mysql_query("update download set accesed=$accest where
bokid=$bokid");
```

4.3. Modulul de căutare

Orice sit Web cu un conţinut bogat trebuie să posede şi un modul de căutare pentru a uşura regăsirea informaţiilor. Biblioteca noastră include şi ea un astfel de modul, foarte util – de exemplu – atunci când doriţi să căutaţi o anumită carte al cărei titlu îl ştiţi doar parţial. Modulul pe care-l vom prezenta poate fi folosit cu succes pe orice sit. Căutarea se poate realiza după o expresie, după categorie şi după subcategorie, micşorând în acest fel domeniul de căutare.





Modulul de căutare

Să vedem cum funcţionează acest modul de căutare. Cel mai simplu este să introduceţi cuvintele după care se face căutarea în câmpul **Containing Text**, iar la apăsarea butonului Go cărţile care au în titlu cuvintele căutate vor fi listate. În cazul în care selectaţi o categorie, va apărea automat în câmpul următor o listă din care puteţi alege o subcategorie a acelei categorii. Dacă veţi apăsa acum butonul **Go**, vor fi afişate toate cărţile din acea subcategorie, iar dacă vreţi să căutaţi cărţile dintr-o anumită subcategorie şi care conţin în titlu anumite cuvinte va trebui să alegeţi categoria, apoi subcategoria şi să introduceţi acele cuvinte în câmpul **Containing Text**. Cărţile care respectă acest criteriu vor fi listate.

Iată în continuare codul PHP realizând căutarea:

```
<script language="JavaScript" type="text/javascript">
function func()
{ // apelata la modificarea unei optiuni
  // din lista <select>
  str = "<?php echo $PHP_SELF;?>?fcatid=" +
        document.search.fcatid.value +
        " & catid = <?php echo $catid;?> &" +
        " subcatid=<?php echo $subcatid; ?>";
  window.location = str;
}
</script>
```



```
<form name="search" action=" <?php
echo("$PHP_SELF?subcatid=$subcatid&catid=$catid&fcatid=$fcatid&
where=$where") ?>" method="post">
<font color="#335588">By Category:</font>

<select name="fcatid" size="1" onchange="func();">
  <option selected value="">
  <?php
    if ($fcatid > 0)
    {
       $ffcat = $fcatid;
    }
    if ((isset($fcatid)) and ($fcatid != 10)
         and ($fcatid > 0))
    {
      $categ = mysql_query("select catid, name from
categories where catid=$fcatid");
      $cat = mysql_fetch_array($categ);
      $fcatid = $cat["catid"];
      $name = $cat["name"];
      echo("$name");
      echo("<option value=10>any category\n");
    }
    else
    {
      echo("Any Category");
      $fcatid = 10;
    }
    $categ = mysql_query("select catid, name from
categories where catid!=$fcatid");
    while ($cat = mysql_fetch_array($categ))
    {
      $fcatid = $cat["catid"];
      $name = $cat["name"];
      echo("<option value='$fcatid'>$name\n");
    }
    if((isset($ffcat)) and ( $ffcat>0) and ($ffcat!=10))
    {
      $subcateg  = mysql_query("select subcatid, sname
from subcat where scatid = $ffcat");
      $where = " where cat = $ffcat and public>$public";
    ?>
    </select><br />
    <font color="#335588">By Subcategory:</font>
    <select name="fscatid" size="1">
```

```
    <option selected value="">Any Subcategory
    <?php
      while ($scat = mysql_fetch_array($subcateg))
      {
        $fscatid = $scat["subcatid"];
        $sname = $scat["sname"];
        echo("<option value='$fscatid'>$sname\n");
      }
    }
  ?>
</select><br />
<font color="#335588">
Containing Text: </font>
<input type=text name="searchtext" size="19">
<!-- butonul imagine de tip submit -->
<input style="position:absolute;top:260px" border="0"
       src="resurse/go.gif" name="submit"
       value="Search" type="image" >
</form>
```

Să analizăm şi aici secvenţa de cod de mai sus. Funcţia JavaScript `func()` este apelată de fiecare dată când selectăm o categorie şi repopulează lista subcategoriilor conform categoriei selectate. Variabila `$fcatid` păstrează mereu indicele opţiunii alese din lista categoriilor, iar dacă nu a fost aleasă nici o categorie, are valoarea implicită 10 echivalentă cu **Any Category**.

```
<select name="fcatid" size="1" onchange="func();">
  <option selected value="">
  <?php
    if($fcatid > 0)
    {
      $ffcat = $fcatid;
    }
    if((isset($fcatid)) and ($fcatid != 10) and
        ($fcatid > 0))
    {
      $categ = mysql_query("select catid, name
from categories where catid=$fcatid");
      $cat = mysql_fetch_array($categ);
      $fcatid = $cat["catid"];
      $name = $cat["name"];
      echo("$name");
      echo("<option value=10>Any Category\n");
    }
    else
```



```
    {
      echo("Any Category");
      $fcatid = 10;
    }
```

Această secvenţă de cod construieşte lista de tip *drop-down* conţinând categoriile. Verificăm dacă am selectat vreo subcategorie (`if((isset($fcatid))`) şi dacă nu am selectat opţiunea implicită **Any Category**, iar dacă ambele condiţii sunt îndeplinite, înseamnă că este selectată o categorie şi trebuie să afişăm pe prima poziţie din listă numele acesteia (`echo("$name");`). Imediat după numele categoriei selectate vom afişa opţiunea implicită **Any Category**, cu valoarea 10 (`echo("<option value=10> Any Category\n");`). Dacă nu am selectat nici o categorie, atunci prima din listă va fi opţiunea implicită. În continuare populăm lista cu categoriile rămase (`catid!= $fcatid`).

```
    if((isset($ffcat)) and ( $ffcat > 0)
        and ($ffcat != 10))
    {
      $subcateg  = mysql_query("select subcatid, sname
from subcat where scatid = $ffcat");
      $where = " where cat = $ffcat and public>$public";
```

Am spus mai sus că vom discuta despre variabila `$where` în cadrul modulului de căutare, şi iată că a sosit momentul. Pentru a realiza căutarea în baza de date după un anumit criteriu trebuie să folosim clauza `where`. Până în acest moment singura condiţie pe care trebuie s-o respecte cărţile căutate este să facă parte din categoria selectată (`$where = " where cat = $ffcat`).

Secvenţa de cod rămasă realizează următoarele operaţii: popularea listei cu subcategorii, crearea câmpului în care se vor introduce cuvintele după care se face căutarea şi crearea butonului Go.

Până acum am prezentat partea de cod care se ocupă în special de partea de interfaţă a modulului de căutare. În continuare urmează codul-sursă care formează condiţiile după care se va realiza căutarea (sub forma unor şiruri de caractere, al căror conţinut va fi ulterior înglobat în diverse comenzi):

```
<?php
    if( isset($fcatid) )
    {
      $select = "select distinct bookid, title, author,
cat, subcat, date, link, type";
      $from   = " from books";
      if ( $where == "" )
      {
      $where  = " where bookid != 0 and public>$public";
      }
      if ($fcatid != "" and $fcatid != 10 and $fcatid > 0)
```

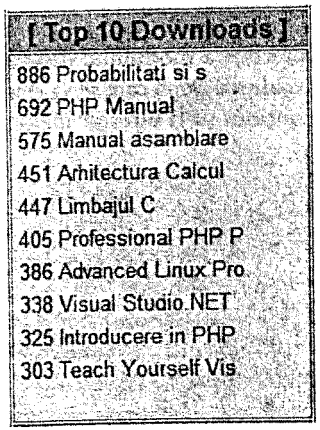
```
        { // este selectata o categorie
          $where .= " and cat=$fcatid and public>$public";
        }
        if ($fscatid != "" and $fscatid > 0)
        { // este selectata o subcategorie
          $where .=
            " and subcat=$fscatid and public>$public";
        }
        if($ff != "" and $ff > 0)
        {
          $where .= " and cat=$ff and public>$public" ;
        }
      }
      $booklist = mysql_query($select . $from . $where . "
and public>$public and title like '%$searchtext%' order by
date desc ");
?>
```

Această secvenţă de cod se va afla la începutul fişierului `programming.php`. Interogarea bazei de date după care se va realiza căutarea va fi formată prin concatenarea acestor trei variabile: `$select.$from.$where`, operaţia de concatenare fiind marcată prin semnul „.". Dintre acestea, cea care poate avea diferite forme este variabila `$where` deoarece clauza `$where` din comanda SQL `select` poate determina doar căutarea cărţilor al căror titlu conţine anumite cuvinte (`title like '%$search text%'`) sau căutarea după o anumită categorie (`and cat=$fcatid`), sau numai după o anumită subcategorie (`and subcat=$fscatid`), sau căutarea după toate trei condiţiile la un loc.

4.4. Topul celor mai accesate cărţi

Să presupunem că un utilizator doreşte să-şi lărgească orizontul cunoştinţelor din domeniul informaticii, fără a cunoaşte *a priori* titlul cărţilor pe care doreşte să le parcurgă. Această persoană poate alege o carte din topul celor mai descărcate cărţi, fiind asigurată de calitatea conţinutului acelei cărţi tocmai datorită numărului mare de descărcări.

Vom genera astfel un top al celor mai accesate cărţi. Tot ceea ce trebuie să realizăm din punctul de vedere al implementării este să selectăm cărţile în ordine descrescătoare după numărul de descărcări (*download*-uri) şi să afişăm doar primele 10 dintre acestea. Pentru fiecare carte vom afişa numărul de descărcări şi doar primele 18 caractere din titlu pentru



Topul celor mai descărcate cărţi

a putea încăpea în celula alocată. Fiecare titlu de carte reprezintă şi o legătură pentru descărcarea acelei cărţi, uşurând în acest fel munca utilizatorului care nu mai trebuie să caute acea carte.

Vom expune doar secvenţa de cod care realizează selectarea cărţilor din baza de date, afişarea nefiind complicată:

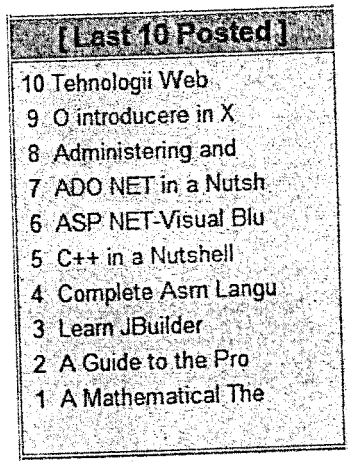
```
<?php
    $inc = 10;
    $ddd = mysql_query("select bokid,accesed from download
order by accesed desc limit 0, 10");
    while ($dddd = mysql_fetch_array($ddd))
    {
      $bbid = $dddd["bokid"];
      $top = mysql_query("select bookid, title, link, cat,
subcat, public from books where public>$public and
bookid=$bbid limit 0, 10");
      $topp = mysql_fetch_array($top);
      $title2 = $topp["title"];
      $acc = $dddd["accesed"];
      $cat = $topp["cat"];
      $subcat = $topp["subcat"];
      $link = $topp["link"];
      $bookid = $topp["bookid"];
      $ppublic = $topp["public"];
      $accesed = $acc;
      $title3 = substr($title2, 0, 18);
      // ...
    }
?>
```

Similar se realizează şi lista celor mai recente 10 cărţi introduse în baza de date, singura diferenţă fiind aceea că selectarea descrescătoare se va efectua după dată şi nu după numărul de descărcări, iar la afişare – în loc de numărul de accesări – vom afişa o numerotare care descreşte de la valoarea 10 până la valoarea 1. Această listă poate fi utilă vizitatorilor grăbiţi care vor să consulte rapid ce a apărut nou pe sit.

Iată codul care realizează selectarea celor mai recente 10 cărţi introduse în bibliotecă:

```
<?php
    $ind = 10;
    $top = mysql_query("select bookid, title, link, cat,
subcat from books where public>$public order by date desc
limit 0, 10");
    while ($topp = mysql_fetch_array($top))
    {
      $title2 = $topp["title"];
```

```
        $cat = $topp["cat"];
        $subcat = $topp["subcat"];
        $link = $topp["link"];
        $bookid = $topp["bookid"];
        $ddown = mysql_query("select accesed from download
where bokid=$bookid");
        $dwn = mysql_fetch_array($ddown);
        $accesed = $dwn["accesed"];
        $acc = $dwn["accesed"];
        $title3 = substr($title2, 0, 18);
        // ...
    }
?>
```



Lista celor mai recente 10 cărţi adăugate în bibliotecă

5. La sfârşit de capitol

Pentru a concluziona, putem afirma că informaţiile acumulate în acest capitol pot fi folosite pentru a realiza orice aplicaţie care foloseşte limbajul PHP şi sistemul de baze de date MySQL, după cum se va vedea şi din parcurgerea următoarelor capitole ale cărţii. Avizăm cititorul că aplicaţiei i se pot adăuga multe alte facilităţi, dintre acestea putându-se menţiona:

- conectarea pe baza unui nume de utilizator şi a unei parole (a se vedea şi cele descrise de capitolele 2 şi 3);
- un modul de căutare îmbunătăţit;
- o grafică mai plăcută (a se vedea utilizarea graficii 2D şi 3D din capitolul 5 sau generarea de fişiere grafice prezentată de capitolul 6 al volumului de faţă);
- posibilitatea ca biblioteca să fie accesibilă – de ce nu? – de pe dispozitive mobile (a se vedea capitolul 4);
- fiecare carte să dispună de o opţiune prin care un utilizator să-i poată da o notă (de la 1 la 10, de exemplu) şi să-şi spună părerea despre carte – dar numai după ce a citit-o în prealabil (un sistem de vot este prezentat în următorul capitol);
- un *chat* sau un forum unde utilizatorii să poată discuta pe diferite teme din informatică (să nu uităm că acest sit Web se doreşte a fi un portal educaţional şi nu doar o simplă bibliotecă);
- un modul mai facil de inserare a cărţilor, prin care oricine, pe baza unui nume de utilizator şi a unei parole, să poată adăuga documentaţii în bibliotecă, eventual folosind facilităţile oferite de un serviciu Web bazat pe XML (a se parcurge capitolul 7);
- adăugarea unei noi categorii de resurse, şi anume prezentările video, acestea reprezentând, după părerea autorului, viitorul în ceea ce priveşte autoeducaţia.

Comentarii bibliografice

O privire de ansamblu asupra serverului de aplicaţii PHP se poate face consultând articolul lui Mircea Sârbu, „PHP: Hypertext Preprocessor", apărut în *NET Report* (august 2000) şi partea G din volumul II al cărţii *Tehnologii Web* scrisă de Sabin Buraga. A se vedea şi anexa B a cărţii de faţă.

Există o serie de materiale dedicate subiectului PHP, dintre care vom enumera doar câteva (în ordinea alfabetică a numelor autorilor):

- Toby Butzon, *PHP by Example*, QUE, 2001
- Ken Egervari, Jon Parise, John Coggershall, *Professional PHP 4 Programming*, Wrox Press, 2002
- Rasmus Lerdorf, Kevin Tatroe, Bob Kaehms, Ric McGredy, *Programming PHP*, O'Reilly & Associates, 2002
- Bill McCarty, *PHP 4*, Teora, Bucureşti, 2002

Desigur, cititorii sunt invitaţi să parcurgă următoarele capitole ale cărţii, în care se vor oferi diverse alte detalii privitoare la programarea PHP şi la realizarea de alte tipuri de aplicaţii Web.

Aplicaţia *PHPTriad* amintită în text este disponibilă gratuit la adresa **http://sourceforge.net/projects/phptriad**, iar varianta ei îmbunătăţită poate fi regăsită sub denumirea comercială de *Sokkit* (a se vedea situl **http://www.sokkit.net/**).

CAPITOLUL 2

TopWeb – lupta pentru voturi

Constantin Puşcaşu, puco@infoiasi.ro

Pe parcursul acestui capitol vor fi expuşi paşii necesari pentru realizarea unui sit care gestionează topuri ale accesărilor de situri Web. Se folosesc limbajul de programare PHP şi serverul de baze de date MySQL. Vor fi prezentate toate detaliile necesare pentru crearea bazei de date, precum şi principalele tehnici folosite în procesul de contorizare a voturilor primite de pe siturile utilizatorilor.

1. Prezentare generală

Ultimii ani au marcat o creştere impresionantă a spaţiului WWW, fapt ce a dus la necesitatea dezvoltării unor instrumente puternice de căutare şi catalogare a informaţiei disponibile *on-line*. Printre acestea, primul loc îl ocupă motoarele de căutare, însă – din păcate... –, adesea rezultatele returnate de acestea nu oferă precizări consistente asupra calităţii informaţiei găsite sau, mai rău, returnează multe rezultate irelevante.

O alternativă foarte eficientă la motoarele de căutare o constituie siturile de tip *top*, care oferă vizitatorilor liste cu cele mai bine cotate (de către vizitatori) situri dintr-o anumită categorie sau la nivel global.

Folosirea acestui tip de serviciu presupune existenţa pe fiecare dintre siturile avute în vedere a unei zone (de obicei o imagine) unde vizitatorii pot da *click* pentru a vota situl respectiv, având astfel posibilitatea să-şi exprime aprecierea faţă de calitatea sitului respectiv.

De ce se bucură acest tip de situri de un real succes? Pentru a răspunde obiectiv la această întrebare, trebuie să analizăm situaţia atât din perspectiva proprietarilor de situri, cât şi din perspectiva vizitatorilor.

Dacă deţineţi un sit şi folosiţi un astfel de serviciu, cu cât aveţi mai multe voturi, situl dumneavoastră va urca în top şi, implicit, veţi avea mai mulţi vizitatori. De asemenea, veţi avea la dispoziţie instrumente de analiză a traficului pe situl dumneavoastră (numărul de vizitatori, voturi, *browser* folosit etc.). Pe scurt, acest tip de serviciu reprezintă o metodă gratuită de promovare a sitului.

Să privim acum lucrurile din perspectiva utilizatorului de Internet: atunci când doreşte să viziteze un sit românesc de cultură, spre exemplu, este mult mai comod să acceseze secţiunea corespunzătoare din cadrul unui sit top şi să vadă lista celor mai vizitate situri din această categorie decât să folosească un motor de căutare care va returna multe rezultate irelevante.

Există un singur dezavantaj pentru siturile de tip top, şi acesta este de ordin psihologic, şi nu tehnic: este foarte posibil ca vizitatorii siturilor Web promovate să fie iritaţi de faptul că sunt rugaţi (sau chiar obligaţi) să voteze pentru ele, riscându-se astfel şi erodarea de imagine pentru situl de top.

În concluzie, ambele părţi (vizitatorii şi siturile) au de câştigat din utilizarea unui astfel de serviciu atât timp cât siturile nu forţează limita bunului-simţ pentru a acumula mai multe voturi.

Pentru a testa funcţionalitatea aplicaţiei analizate în acest capitol puteţi accesa o variantă demonstrativă la una din următoarele adrese:

- **http://students.infoiasi.ro/~puco/lic**
- **http://www1.webartwiz.com/~pucsin/lic**

Arhitectura generală a sistemului

În această secţiune va fi expusă la nivel conceptual arhitectura unui sistem de centralizare a voturilor provenite de la utilizatori. Vor fi prezentate componentele esenţiale ale sistemului, precum şi interacţiunile dintre diverse module.

Pentru stocarea informaţiilor referitoare la utilizatori, voturi, vizitatori şi statistici va fi folosită o bază de date, de al cărei management se va ocupa serverul MySQL. Detaliile despre structura şi construirea bazei de date, precum şi argumentele care au dus la alegerea serverului MySQL, vor fi expuse ulterior în cadrul acestui capitol.

În primul rând, pentru funcţionarea sitului este absolut necesar un sistem de management al utilizatorilor, care să permită crearea de conturi (înregistrarea utilizatorilor noi), accesul utilizatorilor deja înregistraţi la informaţiile referitoare la contul lor, precum şi modificarea acestor detalii şi salvarea lor. Acest sistem trebuie să aibă un grad sporit de securitate pentru a nu permite accesul utilizatorilor neautorizaţi la datele altor utilizatori. Autentificarea se va face într-un mod clasic, prin intermediul unui nume de utilizator şi al unei parole.

Modulul esenţial al unei astfel de aplicaţii este cel care se ocupă de recepţionarea şi validarea voturilor de pe siturile utilizatorilor. Pentru acest modul este vitală securitatea. Este esenţial să fie ignorate voturile care provin în mod repetat de la acelaşi vizitator (identificat în mod unic prin intermediul adresei IP a calculatorului-client de pe care trimite votul). De asemenea, este vital să fie implementat un mecanism care să împiedice votarea automată (de exemplu, voturile declanşate automat, prin intermediul unor programe JavaScript, care vor conduce la creşterea artificială şi necinstită a numărului de voturi înregistrate pentru un sit).

Fiecare vot va fi înregistrat în baza de date, împreună cu detaliile aferente (data la care a fost înregistrat votul, de la ce adresă IP provine votul, precum şi cărui sit aparţine).

O altă componentă importantă a sistemului este modulul de contorizare a detaliilor despre vizitatorii şi numărul de afişări ale paginilor unui sit. Aceste informaţii vor fi utilizate pentru elaborarea statisticilor aferente fiecărui sit Web în parte.

Important este, pentru un asemenea sit, şi mecanismul care realizează şi actualizează topul. Acest modul generează liste cu siturile membre, liste care vor putea fi ordonate după numărul de voturi (vizite) înregistrate pentru un sit în ziua, săptămâna sau luna curentă.

Aceasta este partea sitului accesibilă public, ordinea siturilor înregistrate reflectând ordinea preferinţelor vizitatorilor care au accesat respectivele situri. Această secţiune a sitului va fi accesată de vizitatori cu scopul de a găsi cele mai bune situri dintr-o anumită categorie. De asemenea, secţiunea va fi accesată (cu emoţie) frecvent şi de către proprietarii siturilor membre pentru a afla pe ce loc este situat situl lor. Pentru a menţine relevanţa listelor prezentate, ele vor trebui actualizate cât mai des, pentru a fi în concordanţă cu voturile înregistrate. Mecanismul de actualizare a topului va fi expus mai jos.

Aceasta este, pe scurt, arhitectura generală a unui sit de tip top. În continuare va fi detaliat fiecare modul al sistemului. Pentru înţelegerea sub-capitolelor următoare sunt necesare diverse cunoştinţe elementare de *HTML* (*HyperText Markup Language*), *PHP: Hypertext Processor*, *SQL* (*Structured Query Language*) şi *MySQL* (a se vedea şi bibliografia generală a cărţii de faţă).

2. Baza de date

2.1. Introducere

Un sit de tip top acumulează în timp o cantitate semnificativă de date referitoare la membri, voturi, vizitatori, statistici. Pentru a fi asigurat accesul rapid la aceste date este inevitabilă folosirea unui server de baze de date care să se ocupe de managementul informaţiei.

Am ales serverul de baze de date MySQL pentru că este rapid, sigur şi – foarte important – *open source*, adică poate fi folosit gratuit (pentru mai multe informaţii referitoare la condiţiile de utilizare vizitaţi **http://www.mysql.com/products/mysql/index.html**).

Este recunoscut faptul că MySQL este unul dintre cele mai rapide servere SQL, dacă nu chiar cel mai rapid. Rapiditatea sa este determinată în bună măsură de faptul că o serie dintre facilităţile specifice serverelor profesionale de baze de date nu sunt prezente în versiunea standard (suport pentru tranzacţii, proceduri stocate, chei străine etc.).

Lipsa acestor facilităţi nu reprezintă însă un handicap pentru serverul MySQL, ci un avantaj. Se poate afirma că MySQL este un server de baze de date specializat pentru aplicaţii Web care nu necesită un grad maxim de securitate a datelor. Marea majoritate a aplicaţiilor Web funcţionează excelent fără tranzacţii. Bineînţeles că o aplicaţie Web bancară va avea nevoie de facilităţi avansate care nu pot fi oferite de MySQL. În această situaţie putem recurge la alte sisteme de management avansat al

bazelor de date precum PostgreSQL (disponibil liber, principalul „adversar" al serverului MySQL) sau Oracle (o sofisticată soluţie comercială, nu chiar foarte ieftină).

Implementarea iniţială a serverului MySQL s-a făcut într-un context în care viteza de execuţie trebuia să fie prioritară, în defavoarea tranzacţiilor. Ar fi fost inutil să se ofere suport pentru tranzacţii dacă serverul ar fi devenit atât de lent încât să fie practic inutilizabil pentru operaţiile unde am fi avut nevoie de timp de reacţie minim.

Astfel putem înţelege de ce este MySQL cel mai utilizat server de baze de date pentru aplicaţii Web. Dovadă a eficienţei, stabilităţii, scalabilităţii şi siguranţei acestuia stau numeroasele portaluri Web gigant care îl folosesc cu succes. Un exemplu elocvent pentru a susţine afirmaţia anterioară este arhicunoscutul portal *Yahoo!* (**www.yahoo.com**) care, după cum se ştie, este folosit de milioane de utilizatori.

MySQL este un produs bazat pe paradigma *client-server*. Ca urmare, este nevoie de un **nume de cont** şi de o **parolă** pentru fiecare client în parte în vederea accesării autentificate a serverului care ne va găzdui bazele de date.

2.2. Crearea tabelelor

Pentru a crea tabele veţi avea nevoie de un **nume de cont** şi de o **parolă** pe un server MySQL. De asemenea, va fi necesar să aveţi creată o bază de date care va conţine tabelele sitului. Aceste operaţii pot fi făcute fără probleme folosind o *consolă* MySQL (terminal în mod text prin intermediul căruia puteţi rula comenzi SQL) sau un program ce beneficiază de o interfaţă grafică, program care se conectează la serverul MySQL şi permite, de asemenea, rularea de comenzi SQL pe server. În acest sens puteţi folosi *phpMyAdmin* sau *MyCC* (*MySQL Control Center*). Mai multe informaţii referitoare la aplicaţia MyCC puteţi găsi pe situl **http://www.mysql.com**. A se vedea şi capitolul precedent.

Autorul a considerat că este mult mai practic să folosească suportul de comunicare cu MySQL al limbajului PHP pentru a prezenta crearea bazei de date şi a tabelelor. Astfel, cititorul se va familiariza cu funcţiile PHP care permit rularea interogărilor MySQL din cadrul scripturilor PHP.

Pentru început va trebui să ne conectăm la serverul MySQL. Scriptul PHP de mai jos realizează conexiunea cu un server MySQL care rulează pe aceeaşi maşină ca şi serverul Web (*localhost*).

```
<?php
  $handler = mysql_connect("localhost", "nume_cont",
             "parola_cont") or die("Nu m-am putut conecta!");
  echo "Ne-am conectat la serverul MySQL localhost";
  mysql_close($handler);
?>
```

Astfel, funcţia *mysql_connect()* încearcă să stabilească o conexiune cu serverul *localhost* (de pe maşina locală) pentru utilizatorul cu numele *nume_cont*, care are parola *parola_cont*.

Dacă stabilirea conexiunii eşuează, va fi afişat mesajul *„Nu m-am putut conecta!"*. După cum probabil aţi intuit, funcţia *mysql_close()* închide conexiunea cu serverul de baze de date.

Să creăm în continuare baza de date top_web conţinând tabelele care ulterior vor fi folosite:

```
<?php
  $handler = mysql_connect("localhost", "nume_cont",
             "parola_cont") or die("Nu m-am putut conecta!");

  mysql_query("CREATE DATABASE top_web") or
     die("Nu am putut crea baza de date!");

  mysql_close($handler);
?>
```

Funcţia PHP *mysql_query()* permite rularea unei interogări SQL pe serverul de baze de date. Putea fi folosită, de asemenea, funcţia PHP *mysql_create_db ("nume_baza_date")*. Folosirea acestei funcţii este considerată ineficientă şi demodată. În schimb, este folosită foarte intens (acolo unde este posibil) funcţia *mysql_query()*.

Notă: În general, pe un server public, contul MySQL pe care îl veţi deţine nu va avea suficiente privilegii pentru a vă permite crearea unei noi baze de date. În mod obişnuit, veţi avea la dispoziţie o singură bază de date în care să vă creaţi tabelele.

Următorul pas este selectarea bazei de date. Această operaţie are următoarea semnificaţie: toate interogările pe care le veţi rula ulterior vor afecta tabelele conţinute în baza de date selectată, iar tabelele pe care le veţi crea vor fi conţinute, de asemenea, în baza de date selectată. Scriptul PHP de mai jos exemplifică această operaţie.

```
<?php
  $handler = mysql_connect("localhost", "nume_cont",
             "parola_cont") or die("Nu m-am putut conecta!");

  mysql_select_db("top_web") or
    die("Selectarea bazei de date top_web a esuat!");

  mysql_close($handler);
?>
```

O dată selectată baza de date vom putea crea tabelele necesare în cadrul sitului Web.

În primul rând trebuie să decidem de ce tabele avem nevoie pentru a stoca informaţiile. Poate cea mai importantă tabelă a sistemului este cea care stochează informaţiile despre membri. Aici vor fi memorate numele utilizatorilor, parola acestora, precum şi informaţii referitoare la situl lor (nume, URL, descriere). Vom denumi această tabelă **conturi**.

Scriptul PHP de mai jos va crea tabela **conturi**.

```
<?php
  $handler = mysql_connect("localhost", "nume_cont",
             "parola_cont") or die("Nu m-am putut conecta!");

  mysql_select_db("top_web") or
    die("Selectarea bazei de date top_web a esuat!");

  mysql_query("CREATE TABLE conturi (
      ID mediumint(7) unsigned NOT NULL auto_increment,
      Parola char(16) NOT NULL,
      Nume varchar(100) NOT NULL,
      Email varchar(120) NOT NULL,
      Nume_sit varchar(100) NOT NULL,
      URL_sit varchar(120) NOT NULL,
      Descriere_sit varchar(200) NOT NULL,
      Data_creare_cont DATE NOT NULL,
      Categorie smallint(5) unsigned NOT NULL,
      Subcategorie smallint(5) unsigned NOT NULL,
      Limba varchar(50) NOT NULL DEFAULT 'Romana',
      Banner varchar(120) NOT NULL,
      PRIMARY KEY (ID),
      UNIQUE KEY nodouble(Email)
      ) TYPE=MyISAM") or die(mysql_error());

  mysql_close($handler);
?>
```

În lista următoare sunt prezentate detaliile referitoare la câmpurile tabelei **conturi**.

- **ID** – va conţine un număr unic (unicitate asigurată de proprietatea *auto_increment* şi de faptul că acest câmp este cheie primară) care va identifica un utilizator. Acest ID va fi folosit de către utilizator pentru a accesa contul său.
- **Parola** – acest câmp va conţine parola utilizatorului în formă criptată. Astfel, numai utilizatorul va cunoaşte parola de acces în contul său. În cazul în care utilizatorul uită parola, sistemul va genera o parolă nouă ce va fi trimisă într-un *e-mail* la adresa utilizatorului. Pentru criptarea parolei va fi folosită funcţia MySQL *PASSWORD()*, care returnează un număr hexazecimal cu o lungime de 16 caractere. Această funcţie foloseşte un algoritm de criptare ireversibil, asemănător cu cel adoptat de sistemul de autentificare al sistemelor de operare UNIX/Linux.
- **Nume** – acest câmp va conţine numele şi prenumele utilizatorului.
- **Email** – adresa de *e-mail* folosită de utilizator pentru crearea contului. Pentru a ne asigura că vor fi înregistraţi doar utilizatori cu adrese de *e-mail* valide, utilizatorul va primi după înregistrare un *e-mail* cu o parolă generată automat, parolă absolut necesară pentru accesarea contului. De asemenea, nu se vor putea crea mai multe conturi care să folosească aceeaşi adresă de *e-mail*.

- **Nume_sit** – denumirea sitului utilizatorului. Această denumire va fi afişată în cadrul listelor topului.
- **URL_sit** – adresa URL (*Uniform Resource Locator*) la care poate fi accesat situl utilizatorului.
- **Descriere_sit** – descrierea sitului utilizatorului, care va fi, de asemenea, afişată în cadrul listelor topului.
- **Data_creare_cont** – câmp de tip dată calendaristică reţinând data la care a fost creat contul. Acest câmp va fi folosit pentru a identifica conturile care trebuie şterse (dacă un cont nu este activat în decurs de 3 zile, va fi şters automat de către aplicaţie).
- **Categorie** – codul categoriei de situri din care face parte situl utilizatorului. Denumirea categoriei va fi regăsită în tabela **categorii**.
- **Subcategorie** – codul subcategoriei de situri din care face parte situl utilizatorului. Denumirea subcategoriei, de asemenea, va fi regăsită în tabela **subcategorii**.
- **Limba** – limba (de exemplu, română sau engleză) care va fi folosită la afişarea interfeţei utilizator atunci când utilizatorul va accesa contul său.
- **Banner** – adresa (URI-ul) la care se regăseşte *banner*-ul sitului său. *Banner*-ul este o imagine de dimensiuni 200 x 70 pixeli care va fi afişată în listele topului alături de celelalte informaţii referitoare la situl corespunzător.

După cum am menţionat mai sus, siturile înregistrate în baza de date vor aparţine unei categorii, respectiv subcategorii (spre exemplu, un sit poate aparţine categoriei *Calculatoare* şi subcategoriei *Internet*).

Astfel, ajungem la concluzia că mai sunt necesare încă două tabele, una în care să memorăm categoriile şi cealaltă pentru subcategoriile de situri.

În tabela **conturi** vor fi folosite doar codurile asociate unei denumiri de categorii (subcategorii), şi nu denumirile complete. Vom stabili o relaţie *one-to-many* între tabela **categorii** şi tabela **subcategorii**. Astfel, o subcategorie de situri va putea aparţine doar unei categorii. Scriptul PHP de mai jos conţine interogările SQL care vor crea structura acestor tabele.

```
mysql_query("CREATE TABLE categorii (
   ID smallint(5) unsigned NOT NULL auto_increment,
   Denumire_categorie varchar(50) NOT NULL,
   PRIMARY KEY (ID),
   ) TYPE=MyISAM") or die(mysql_error());

mysql_query("CREATE TABLE subcategorii (
   ID smallint(5) unsigned NOT NULL auto_increment,
   id_categorie smallint(5) unsigned NOT NULL,
   Denumire_subcategorie varchar(50) NOT NULL,
   PRIMARY KEY (ID),
   ) TYPE=MyISAM") or die(mysql_error());
```

Este evident faptul că într-un sit de tip top este esenţial conceptul de vot. Iminent apare necesitatea creării unei tabele în care să fie stocate datele despre voturile primite de la utilizatori. De asemenea, este necesară crearea unei tabele care să conţină datele despre vizitele înregistrate pe siturile înscrise la serviciul de topuri.

Mecanismul este următorul: când un vizitator accesează unul dintre siturile membre, este înregistrată vizita acestuia. În momentul când vizitatorul apasă butonul de vot, pentru validarea votului în primul rând se verifică dacă nu s-a mai înregistrat un vot de la acel IP în ziua curentă. Dacă se trece de acest pas, votul este validat.

Este logic că un vizitator nu ar fi putut vota situl fără să-l fi vizitat. Acestă măsură de securitate filtrează însă eventualele voturi primite de la diverse scripturi CGI de votare automată.

Următorul script PHP prezintă structura tabelelor **voturi** şi **vizite**.

```
mysql_query("CREATE TABLE voturi (
   ID mediumint(7) unsigned NOT NULL,
   IP varchar(16) NOT NULL,
   Data DATE NOT NULL,
   Visit DECIMAL(1,0) DEFAULT 0,
   Vot DECIMAL(1,0) DEFAULT 0
   ) TYPE=MyISAM") or die(mysql_error());

mysql_query("CREATE TABLE vizite (
   ID mediumint(7) unsigned NOT NULL,
   IP varchar(16) NOT NULL,
   Data DATE NOT NULL,
   Afisari smallint(4) DEFAULT 0
   ) TYPE=MyISAM") or die(mysql_error());
```

Câmpul **Vot** (din tabela **voturi**) de tip *boolean* (conţine numai valorile 0 sau 1) va fi setat pe 1 numai dacă există deja o înregistrare în tabelă pentru ziua curentă, de la acelaşi IP care are setat câmpul **Visit** la valoarea 1. Câmpul **ID** conţine numărul din tabela **conturi** care identifică în mod unic un utilizator.

Câmpul **Afişări** din tabela **vizite** va înregistra numărul de afişări ale paginilor sitului utilizatorului cu numărul de identificare **ID**, de la IP-ul conţinut în câmpul **IP**, din data înregistrată în câmpul **Data**.

Având creată baza de date şi tabelele necesare putem trece la implementarea modulelor descrise pe scurt în subcapitolul anterior.

3. Managementul utilizatorilor

3.1. Introducere

În acest capitol va fi detaliat modulul care se ocupă cu înregistrarea şi autentificarea membrilor, precum şi cu editarea şi salvarea datelor referitoare la conturile acestora.

La elaborarea acestui subcapitol s-a pornit de la premisa că cititorul posedă cunoştinţe medii de HTML şi PHP.

3.2. Înregistrarea utilizatorilor

Pentru a beneficia de serviciile acestui sistem de generare a topurilor, un utilizator proprietar de sit trebuie să îşi creeze un cont (să se înregistreze). Pentru înregistrare au fost considerate suficiente următoarele informaţii:

- **Numele** – pentru ca utilizatorul să primească un *e-mail* personalizat, care să i se adreseze personal.
- **Adresa E-mail** – este o informaţie absolut necesară pentru a avea certitudinea că vor fi înregistraţi doar utilizatori cu adrese de *e-mail* valide. Pentru a realiza acest obiectiv, după înregistrarea utilizatorului va fi trimis un mesaj la adresa specificată, mesaj care va conţine o parolă de acces generată automat. Astfel, utilizatorii care au folosit o adresă de *e-mail* falsă, neposedând parola de acces nu vor putea accesa contul creat.

☑ Accept conditiile de utilizare

*Nume Puscasu Constantin

*Email puco@infoiasi.ro

ATENTIE : Adresa de email trebuie sa fie valida.
ID-ul, parola si instructiunile de Login va vor fi trimise via email.

Cont nou

Formularul de înregistrare a utilizatorilor

În scriptul următor este prezentată funcţia care adaugă un utilizator nou.

```
<?php
function adauga_user($nume, $email) {
   // verificam daca mai exista un user inregistrat cu
   // aceasta adresa de e-mail
   $bd = mysql_query("SELECT Email FROM conturi
   WHERE Email='$email'") or die(mysql_error());
   $gasite = mysql_num_rows($bd);
```

```
    // a fost gasit un utilizator cu aceeasi adresa
    if($gasite) return 0;
    // generam parola
    $password = make_password();
    // pregatim informatia pentru adaugarea
    // in baza de date
    $nume = addslashes($nume);
    // adaugam utilizatorul in tabela conturi
    mysql_query("INSERT INTO conturi SET
      Nume='$nume',
      Email='$email',
      Parola='$password',
      Data_creare_cont=CURDATE()") or
        die(mysql_error());
    // aflam ID-ul utilizatorului
    $id_cont = mysql_insert_id();
    // trimitem parola intr-un e-mail
    mail("$email", "Va multumim pentru inregistrare",
    "Stimate $nume,
    Iata informatiile de acces pentru contul
    dumneavoastra:\n
    ID: $id_cont\n
    Parola: $password\n",
    "From: TopWeb <puco@infoiasi.ro>\n
    Content-Type: text/plain");
}
?>
```

Pentru mai multe informaţii despre funcţiile PHP folosite în scriptul de mai sus consultaţi manualul oficial la următoarea adresă: **http://www.php.net/manual/ro**.

3.3. Autentificarea utilizatorilor

În continuare vom analiza câteva generalităţi în legătură cu sistemele de autentificare, modul cum este proiectat cel utilizat de aplicaţia *TopWeb*, detalii de implementare a acestuia şi câteva aspecte legate de securitate.

Un sistem de autentificare presupune permiterea accesului pentru fiecare dintre utilizatori la anumite resurse ale unui sistem, în funcţie de drepturile de acces ale acelui utilizator şi de identitatea acestuia. În general, autentificarea se face pe baza unui nume de utilizator (pentru a stabili identitatea utilizatorului) şi a unei parole de acces la resursele sistemului. Este de la sine înţeles că nu există un sistem de autentificare infailibil, dar poate fi ameliorat prin folosirea unei criptări a datelor cât mai sigură, prin obligarea utilizatorilor să folosescă parole mai lungi (deci şi mai greu de decriptat) etc.

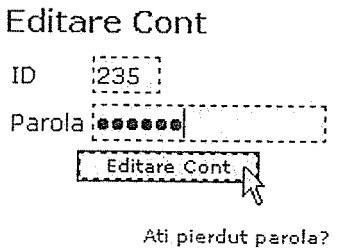
În cazul aplicaţiei de faţă sistemul de autentificare a utilizatorilor permite accesul acestora la o pagină Web în cadrul căreia îşi pot edita datele personale (nume, adresa de *e-mail*, parola, URL-ul sitului înscris în top etc.), precum şi preferinţele referitoare la interfaţa utilizator (*limba* care va fi folosită pentru afişarea textelor interfeţei, precum şi *tema* vizuală ce va fi adoptată pentru formatarea paginilor Web).

Toate informaţiile despre utilizatori (inclusiv datele folosite pentru autentificare), precum şi preferinţele acestora vor fi memorate în tabela **conturi**. Pentru a preveni vizualizarea „accidentală" a parolelor datorită unei configurări precare a serverului de baze de date, vom folosi funcţia de criptare *PASSWORD()*. Acestă funcţie este implementată în MySQL şi permite criptarea unui câmp din cadrul unei tabele, rezultatul fiind un şir de caractere cu lungimea 16. Algoritmul de criptare este ireversibil, diferit de cel folosit pentru criptarea parolelor pe sistemele UNIX. Folosirea acestei funcţii are, de asemenea, avantajul de a economisi memorie (parola memorată în tabelă având doar 16 caractere).

Pentru implementarea sistemului de autentificare va fi folosit sistemul de management al sesiunilor oferit de PHP4. Algoritmul de autentificare a utilizatorilor este următorul:

1. utilizatorul va furniza un ID (număr unic care identifică utilizatorul) şi o parolă de acces;
2. în cazul în care combinaţia ID şi parola nu sunt corecte va fi afişat un mesaj de avertizare, nepermiţând utilizatorului accesul în cont. Dacă datele sunt corecte se continuă cu pasul 3;
3. se înregistrează o variabilă de sesiune care va fi folosită în toate paginile care necesită autentificare, pentru a se verifica dacă un anumit utilizator este conectat;
4. când utilizatorul decide să iasă din cont, vor fi distruse variabila de *login* şi cea de sesiune.

Pentru implementarea sistemului de autentificare vor fi folosite trei funcţii şi anume: *login($ID, $password)*, *is_logged()* şi *logout()*.



Formularul de autentificare a utilizatorilor

Funcţia login($ID, $password)

Această funcţie interoghează tabela în care se găsesc conturile membrilor folosind valorile efective ale parametrilor formali *$ID* (numărul de identificare al utilizatorului

şi *$password* (parola furnizată anterior de utilizator în formularul din pagina Web de conectare – *login*). Dacă nu este găsită o înregistrare conţinând cei doi parametri, funcţia returnează valoarea 0 (nu a fost găsit nici un utilizator care să aibă ID-ul *$ID* şi parola *$password*). În caz contrar, va fi înregistrată variabila de sesiune *$logged*. Deoarece există pericolul ca variabilele de sesiune să fie vizualizate de persoane neautorizate, pentru a spori securitatea este necesar să criptăm informaţiile stocate în acestea.

În consecinţă, variabila *$logged* are o construcţie specială, şi anume: iniţial creăm variabila *$hash=base64_encode(md5($ID.$pass_bd.$secret));* Valoarea variabilei *$secret* este definită în acelaşi fişier în care se va găsi şi definiţia funcţiei *login()*.

Scopul folosirii acestei variabile este sporirea securităţii: în cazul în care ar fi decriptată (foarte puţin probabil), informaţia din variabila de sesiune, şirul rezultat în urma decriptării va fi destul de confuz pentru cititor (decriptor), neputându-se preciza clar care este parola şi care este variabila de mascare. Oricum, parola din acest şir este returnată deja criptată de serverul MySQL cu funcţia *PASSWORD()*. În continuare, este criptat acest şir de caractere folosindu-se funcţia PHP *md5(string str)*. Acestă funcţie aplică un algoritm de criptare ireversibil, valoarea întoarsă fiind un număr hexazecimal de 32 de caractere (de exemplu: *1250dac1f0ec17bb77678f574ca11fc3*). Pentru a ascunde şi acest număr este apelată funcţia *base64_encode(string str)*, care returnează şirul de caractere dat ca argument codificat în baza 64 (de exemplu: *MTI1MGRhYzFmMGVjMTdiYjc3Njc4ZjU3NGNhMTFmYzM=*). Variabila *$logged* se obţine concatenând variabilele *$ID* şi *$hash*.

În final, în sesiune, variabila *$logged* va fi înregistrată în următoarea formă : *7:MTI1MGRhYzFmMGVjMTdiYjc3Njc4ZjU3NGNhMTFmYzM=*

Implementarea în PHP a funcţiei *login()* este descrisă în cele ce urmează:

```
<?php
$secret="variabila de siguranta";

function login($ID, $password) {
  global $secret; // variabila globala

  $handler = mysql_query("SELECT Parola
    FROM conturi WHERE ID='$ID'
    AND Parola=PASSWORD('$password')")
    or die(mysql_error());

    // numarul de campuri selectate
  $result = mysql_num_rows($handler);

    // informatia de login nu este valida
  if($result != 1) return 0;
  list($pass_bd) =  mysql_fetch_row($handler);
  $hash = base64_encode(md5($ID.$pass_bd.$secret));
```



```
    // construim variabila de autentificare
  $logged = "$ID:$hash";

    // inregistram in sesiune variabila
  $_SESSION["logged"] = $logged;

   // autentificarea s-a facut cu succes
  return 1;
?>
```

Funcţia is_logged()

Această funcţie va fi apelată în toate paginile Web ale aplicaţiei *TopWeb*. În cadrul paginilor care necesită autentificare, apelul funcţiei va decide dacă se permite accesul utilizatorului la respectiva pagină, iar în restul paginilor în urma apelului acestei funcţii vor fi setate limba şi tema interfeţei cu utilizatorul preferate de către acesta, în cazul utilizatorilor înregistraţi.

Pentru a realiza autentificarea se încearcă citirea variabilei *$logged* din sesiune. Dacă nu este setată o asemenea variabilă, funcţia returnează valoarea 0 (nu există un utilizator autentificat), iar în caz contrar se apelează funcţia *explode(string separator, string string [, int limit])*.

Funcţia *explode()* returnează un tablou de şiruri de caractere, şiruri obţinute din şirul iniţial considerându-se în acesta şirul de caractere *separator* drept separator de subşiruri. Şirul *separator* nu va apărea în tabloul returnat. În cazul nostru funcţia *explode()* va fi apelată astfel: *$vars=explode(":", $_SESSION["logged"])*. Se va obţine *$vars[0]=$ID* şi *$vars[1]=$hash*.

Următorul pas va consta în executarea unei interogări SQL asupra tabelei **conturi**, pentru a se încerca selectarea parolei corespunzatoare ID-ului *$vars[0]*. Dacă este găsită o înregistrare care să aibă acest ID, se va recrea variabila *$hash* folosindu-se parola obţinută din baza de date şi variabila *$secret*. Se va testa dacă valoarea variabilei *$hash* obţinută coincide cu valoarea variabilei *$vars[1]*.

Dacă nu este găsită în tabelă o înregistrare care să aibă ID-ul *$vars[0]* sau compararea celor două variabile (*$hash* şi *$vars[1]*) returnează *fals*, funcţia *is_logged()* returnează valoarea 0, ceea ce semnifică faptul că „utilizatorul cu ID-ul *$vars[0]* nu este conectat (autentificat)".

Dacă testul de comparare returnează *true*, funcţia *is_logged()* returnează valoarea 1, acest rezultat fiind echivalent cu afirmaţia „utilizatorul cu ID-ul *$vars[0]* este autentificat şi are acces în cont".

Este posibil să existe opinii care să susţină că serverul va consuma resurse considerabile pentru autentificarea utilizatorilor, ceea ce este fals. Tot procesul de autentificare se „întinde" pe parcursul unui interval de timp de ordinul zecilor de milisecunde.

Se obţine astfel un sistem cu un nivel de securitate ridicat. Practic, dacă utilizatorul nu lasă navigatorul cu o sesiune deschisă, singura cale prin care ar putea să-i fie aflată parola de acces este *prin forţă brută* (folosirea unui generator de parole care să compare

parola generată cu cea criptată). Acest lucru presupune în prealabil aflarea şi decodarea variabilei *$logged* din baza 64 şi aflarea parolei criptate de MySQL din variabila *$hash* tot *prin forţă brută*, datorită folosirii funcţiei *md5()*.

Implementarea funcţiei *is_logged()*:

```
<?php
function is_logged() {
  global $secret;

  // nu exista variabila $logged in sesiune
  if(!isset($_SESSION["logged"])) return 0;

  $vars = explode(":", $_SESSION["logged"]);
  $bd = mysql_query("SELECT Parola FROM conturi
    WHERE ID='$vars[0]'") or die(mysql_error());

  $result = mysql_num_rows($bd);
  // nu exista un utilizator cu ID-ul $vars[0]
  if($result != 1) return 0;
  // parola din tabela (este criptata)
  list($pass_bd) = mysql_fetch_row($bd);

  // reconstruirea variabilei din sesiune
  $hash = base64_encode(md5($vars[0].$pass_bd.$secret));

  // utilizatorul cu ID-ul $vars[0] nu este autentificat
  if($vars[1] != $hash) return 0;
  // utilizatorul $vars[0] este autentificat
  else return 1;
}
?>
```

Funcţia logout()

Această funcţie va fi apelată într-un fişier numit *logout.php*, ea având rolul de a distruge sesiunea de lucru a utilizatorului, rezultatul fiind ieşirea sa din cont. Din acest moment utilizatorul nu va mai avea acces la paginile care necesită autentificare, până la o nouă autentificare.

De asemenea, ieşirea din cont a utilizatorului va avea ca efect afişarea interfeţei folosindu-se limba şi tema implicită.

Funcţia constă în ştergerea valorii variabilei *$hash* din variabila *$_SESSION["logged"]*, apelul funcţiei standard *session_destroy()*, apel ce are ca rezultat ştergerea variabilelor înregistrate în sesiune şi, în final, ştergerea *cookie*-ului sesiunii de pe calculatorul utilizatorului.

Implementarea funcţiei *logout()*:

```
<?php
function logout() {
  $_SESSION["logged"] = "";
  session_destroy();

  // dupa distrugerea sesiunii cookie-ul de pe
  // calculatorul utilizatorului persista inca
  // de aceea este necesar sa-l stergem explicit
  setcookie(session_name(), "", "", "/");
}
?>
```

Notă: Cei care posedă cunoştinţe de PHP se vor întreba de ce nu am folosit funcţia *unset($variabila)* pentru ştergerea variabilei *$logged*.

Explicaţia este următoarea: din experienţa de programator am observat o serie de *bug*-uri în interpretorul limbajului PHP (motorul de *scripting* Zend), în sensul că anumite funcţii nu produc rezultatul dorit. Una dintre aceste funcţii este chiar *unset()*. Am remarcat faptul că, în cazul sesiunilor, această funcţie nu funcţionează întotdeauna, variabila continuând să existe, ba chiar mai mult, îşi păstrează valoarea după apelul funcţiei *unset()*.

Din acelaşi motiv, în exemplele anterioare s-a folosit instrucţiunea:

```
$_SESSION["variabila"] = $alta_variabila;
```

în locul instrucţiunilor:

```
$variabila = $alta_variabila;
session_register("variabila");
```

S-au observat şi cazuri în care variabilele înregistrate cu ajutorul funcţiei *session_register()* îşi pierdeau valoarea sau nu se regăseau în sesiune.

Variabila globală *$_SESSION[]* este disponibilă începând cu versiunea PHP 4.0.6. Pentru versiunile mai vechi decât cea amintită puteţi folosi variabila globală *$HTTP_SESSION_VARS[]* ca alternativă pentru funcţia *session_register()*.

3.4. Editarea datelor despre conturile utilizatorilor

În această secţiune va fi expusă partea care se ocupă cu editarea şi actualizarea informaţiilor despre conturile clienţilor.

Un utilizator proprietar de sit poate modifica următoarele date referitoare la contul său:

- categoria din care face parte situl;
- subcategoria de situri căreia îi aparţine situl său;
- limba care va fi folosită la afişarea interfeţei;

- numele sitului său;
- descrierea sitului;
- numele complet al utilizatorului;
- adresa de *e-mail* proprie;
- utilizatorul poate specifica o adresă URL unde se regăseşte afişul publicitar (*banner*-ul) sitului său sau poate face *upload* la *banner* direct pe server;
- de asemenea, utilizatorul are posibilitatea de a schimba parola de acces la contul său.

Am amintit mai sus faptul că utilizatorul poate copia direct pe server *banner*-ul sitului său. Pentru a realiza acest lucru în formularul Web de editare a contului trebuie să existe un câmp de tip *file* şi un câmp *hidden* care se numeşte *MAX_FILE_SIZE* care controlează dimensiunea maximă în număr de octeţi a fişierului care poate fi acceptat pentru acţiunea de *upload*:

```
<input type="hidden" name="MAX_FILE_SIZE"
       value="<?php echo ($MaxBannerSize*1024);?>" />
<input class="input" type="file" name="new_banner" />
```

În continuare este prezentat codul PHP care gestionează copierea fişierului specificat în formularul de mai sus. Pentru a afla numele, respectiv tipul fişierul de transferat se folosesc variabilele globale *$HTTP_POST_FILES[new_banner][name]* şi *$HTTP_POST_FILES[new_banner][type]*.

Pentru a accepta doar fişiere de tip imagine se foloseşte suportul PHP pentru expresii regulate, fiind copiate pe server doar fişierele care au extensia *.gif*, *.jpg*, *.png*. De asemenea, se verifică tipul MIME al fişierului (trebuie să aibă prefixul *image/*).

Se observă că numele fişierului copiat pe server are ca prefix ID-ul utilizatorului. Se recurge la acest artificiu pentru a ne asigura că numele fişierelor din directorul *images/banners/* este unic, evitându-se astfel eventuala suprascriere a unor fişiere.

```
<?php
if ($ID && $member_update &&
   $HTTP_POST_FILES[new_banner][name]) {
      // verificam daca sunt fisiere grafice
   if (preg_match("/\.(gif|jpg|png)$/i",
      $HTTP_POST_FILES[new_banner][name]) &&
      preg_match("/image\//i",
      $HTTP_POST_FILES[new_banner][type])) {
       $new_name = "$ID" .
            "{$HTTP_POST_FILES[new_banner][name]}";
         $path = "images/banners/$new_name";
         // copiem...
         if (copy($new_banner, $path))
         $URL_Banner =
              "$base_url/images/banners/$new_name";
         else
```

```
         echo "Upload-ul a esuat. Dimensiunea maxima
               permisa pentru fisier este $MaxBannerSize Ko";
      }
else
   echo "Upload-ul a esuat. Fisierul trebuie sa fie
         jpg, gif sau png";
}
?>
```

Datele contului pentru: Puscasu Constantin

Categorie:	Calculatoare
Subcategorie:	Linux
Limba:	Romana
Stiri:	Da
Numele sitului: (35 caractere max.)	puco - pagina personala
URL-ul sitului:	http://students.infoiasi.ro/~puco
Descriere: (300 caractere max.)	pagina personala a lui Puscasu Constantin
Tag-uri acceptate: <i><u> 	
Nume:	Puscasu Constantin
Email:	puco@infoiasi.ro
Schimba Parola: (6-30 caractere max.)	
URL-ul banner-ului: (Lasati necompletat daca nu este cazul)	

(Apasati Actualizare pentru a vedea codul HTML pentru situl dumneavostra)

Resetare | Actualizare

Nu aveti setat nici un banner.

Upload-ul de banner-e este limitat la: 30Kb (.gif|.jpg|.png) — D:\webartwiz.jpg — Browse...

Formularul de editare a datelor utilizatorului

Codul HTML (introduceti-l in paginile sitului):

```
<a
href="http://www1.webartwiz.com/~pucsin/lic/vote.php?
ID=23" target="_blank"><img
src="http://www1.webartwiz.com/~pucsin/lic/visit.php?
ID=23" border="0" alt="Votati pentru puco - pagina
personala la TOP Web!" /></a>
```

IMPORTANT: Situl dumneavostra NU va apare in lista daca nu ati activat contul!

Codul HTML pentru utilizatorul cu ID-ul 23

În urma salvării modificărilor, utilizatorul va primi şi codul HTML pe care va trebui să-l insereze în paginile sitului său pentru a putea acumula voturi şi pentru a beneficia de statisticile despre traficul de pe situl său.

4. Contorizarea voturilor

4.1. Introducere

Conceptul fundamental al unei aplicaţii care generează un top este **votul**, pe baza căruia se evaluează aprecierea vizitatorilor pentru un sit. Pentru aplicaţie nu este relevantă obiectivitatea acestor voturi, ci *autenticitatea* lor. Astfel, trebuie împiedicate eventualele tentative de fraudă care ar avea ca rezultat ocuparea unui loc nemeritat în top.

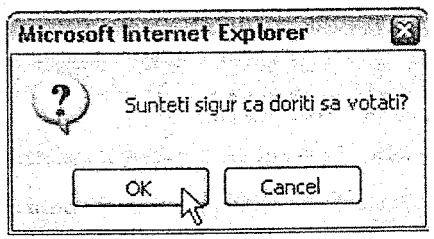
Lista de mai jos prezintă metode pe care utilizatorii le folosesc în încercarea de a obţine mai multe voturi:

1. votarea unui sit de mai multe ori în aceeaşi zi de către acelaşi utilizator;
2. folosirea codului de înregistrare a voturilor pe mai multe situri;
3. inserarea unui script de votare automată;
4. interzicerea accesului vizitatorilor la anumite resurse ale sitului în cazul în care aceştia nu votează;
5. introducerea unor ferestre de tip *pop-up* sau butoane flotante, fapt care „forţează" utilizatorul să voteze (doar prin exprimarea votului se obţine dispariţia controalelor care stresează vizitatorul).

Pentru a combate tentativele de fraudă de la primul punct vom identifica un utilizator la nivel de adresă IP. Astfel, un vot nu va fi validat dacă provine de la o adresă IP de la care s-a mai înregistrat un vot valid în ziua curentă.

Soluţia pentru tentativele de fraudă de la al doilea punct este verificarea variabilei *$HTTP_SERVER_VARS[HTTP_REFERER]* care are ca valoare URL-ul de la care provine votul. Dacă URL-ul din această variabilă nu corespunde cu URL-ul sitului pe care l-a specificat utilizatorul la înregistrare, votul nu va fi luat în considerare.

Pentru a preîntâmpina votarea automată, înainte de înregistrarea votului va fi afişat un dialog de confirmare care va permite vizitatorului să oprească operaţia de validare a votului.



Dialogul de confirmare a unui vot

Metodele de la punctele 4 şi 5 care obligă (mai mult sau mai puţin) utilizatorul să voteze nu pot fi combătute automat. Intră în atribuţiile administratorului să verifice manual dacă există situri care utilizează metodele menţionate. Membrii vor fi avertizaţi la înregistrare, în cadrul condiţiilor de utilizare, că riscă să le fie desfiinţat contul *TopWeb* dacă folosesc astfel de metode pentru a obţine mai multe voturi de la vizitatori.

4.2. Algoritmul de contorizare a voturilor

Detaliile despre voturile înregistrate vor fi memorate în tabela **voturi**. Fiecărei înregistrări îi corespunde o adresă IP, ID-ul membrului căruia îi aparţine votul, data înregistrării, un câmp care conţine o variabilă logică de securitate (*Visit*) care trebuie să aibă valoarea 1 (pentru ca un vot să fie considerat valid trebuie ca situl să fi fost mai întâi vizitat) şi încă un câmp conţinând o variabilă logică (*Vot*). O înregistrare este considerată vot valid dacă ambele variabile logice au valoarea 1.

Funcţionarea sistemului de înregistrare a voturilor are la bază introducerea unei secvenţe de cod HTML (obţinute automat de la aplicaţia *TopWeb*) în paginile sitului membru (este de la sine înţeles că această sarcină revine proprietarului de sit).

Introducerea acestui cod HTML are ca rezultat afişarea unei imagini care are asociată o referinţă către pagina de vot:

```
<a href="http://www.TOPWeb.ro/vote.php?ID=35">...</a>
```

La afişarea imaginii de vot este înregistrată şi vizita utilizatorului pe respectivul sit.

În continuare este prezentat algoritmul sistemului de vot la nivel logic după care va fi expusă şi implementarea în PHP:

- este testată existenţa şi valoarea unei variabile folosite pentru a se decide dacă s-a realizat operaţia de trimitere a formularului din cadrul paginii sau dacă acest formular trebuie generat; valoarea variabilei de test se schimbă după 15 secunde pentru a nu fi validate voturile provenite de la scripturi de votare automată (aceste scripturi fac *submit* automat pentru formularul din cadrul paginii de vot folosind ID-ul respectivului utilizator);
- dacă testul este **true** algoritmul continuă cu pasul următor; dacă nu este setat *cookie*-ul **cookie_test** sau dacă testul returnează valorea fals, este setat *cookie*-ul şi este generat formularul cu datele votului (ID-ul utilizatorului, valoarea de test de mai sus, URL-ul de unde provine votul) şi este trimis automat spre validare;
- se compară valoarea *cookie*-ului cu cea generată. Dacă ID-ul utilizatorului este nul, se consideră că votul a fost anulat. În caz contrar se verifică dacă s-a mai votat de la IP-ul vizitatorului în ultimele 24 ore, se decide validarea votului şi este afişat mesajul corespunzător.

În cele ce urmează vom prezenta implementarea în PHP a algoritmului descris mai sus (fişierul **vote.php**):

```
<?php
// generarea aleatorie a valorilor de test
// sunt schimbate la 15 secunde
```

```
// setarea radacinii generatorului de numere aleatorii
mt_srand((intval(time()/15)*15));
$accept_test = md5(mt_rand(10000,9999999999));
$gate_test   = md5(mt_rand(10000,9999999999));

// prima incarcare a paginii
// trebuie generat formularul de vot si trimis spre validare
if ($gateway != $gate_test || !$gateway) {
// cookie-ul care va fi testat la inregistrarea votului
setcookie("cookie_test", $accept_test);
// generarea formularului
echo "
<form name=\"testgate\" method=\"post\" action=\"vote.php\">
<input style=\"width: 0px;height: 0px\" type=\"text\"
name=\"ID\" />
<input type=\"hidden\" name=\"gateway\" value=\"$gate_test\" />
<input type=\"hidden\" name=\"referer\" value=\"$HTTP_SERVER_
VARS[HTTP_REFERER]\" />
</form>
<script type=\"text/javascript\">";
// se verifica daca vizitatorul vrea sa voteze
echo "
var name = confirm('Sunteti sigur ca doriti sa votati?');
if (name == true) {
  document.testgate.ID.value = '$ID'; }
else {
document.testgate.ID.value = ''; }";
// trimiterea datelor de vot spre validare
echo "document.testgate.submit();</script>"; }

// testul a dat false (variabilele sunt egale) =>
// s-a facut submit la formular
elseif (($gateway == $gate_test) && ($HTTP_COOKIE_VARS
[cookie_test] == $accept_test)) {

// utilizatorul NU a acceptat sa voteze
if($ID == "") goto_page(3);
// utilizatorul a acceptat sa voteze
else {
$query = mysql_query("SELECT URL_sit
  FROM conturi WHERE ID='$ID'") or
  die(mysql_error());
$result = mysql_fetch_array($query);
```



```
// verificam daca votul provine de la
// URL-ul cu care s-a inregistrat
if(!check_referer($result[URL], $referer))
   goto_page(4); // totul este OK
else {
// verificam daca s-a mai votat azi de la IP-ul vizitatorului
$id = $ID;
$query = mysql_query("SELECT * FROM voturi
  WHERE IP='$HTTP_SERVER_VARS[REMOTE_ADDR]'
  AND Data=CURDATE() AND Vot='1'") or die(mysql_error());
$found = mysql_affected_rows();

// in caz ca s-a mai votat azi de la IP
if ($found > 0)
  goto_page(1); // votul este valid
else {
  // verificam daca a fost vizitat azi situl utilizatorului
  // $id de la acest IP
  $query = mysql_query("SELECT * FROM voturi
  WHERE IP='$HTTP_SERVER_VARS[REMOTE_ADDR]'
  AND Data='$var_date' AND ID='$id'") or
  die(mysql_error());
  $found = mysql_affected_rows();
  // daca nu a votat azi dar a vizitat pagina utilizatorului
  if ($found > 0)
  mysql_query("UPDATE voturi SET Vot=1
  WHERE ID='$id' AND IP='$HTTP_SERVER_VARS[REMOTE_ADDR]'
  AND data='$var_date'");
  goto_page(2);
}
} // sfarsit else test URL
} // sfarsit else ID!=""
} // sfarsit elseif


// functia de redirectare si de
// afisare a mesajului corespunzator
function goto_page($status) {
  // selectarea mesajului
  switch ($status) {
    case 1:
     err_pop('S-a votat deja astazi de la acest IP!');
     break;
    case 2:
```

```
      err_pop('Multumim pentru votul dumneavostra!');
      break;
     case 3:
      err_pop('Cererea de anulare a votului
      a fost inregistrata!');
      break;
     case 4:
      err_pop('Votul nu este valid - tentativa
      de frauda a membrului TopWeb');
      break;
   }
// redirectarea la pagina de index a TopWeb
echo "
<script type=\"text/javascript\">
<!--
top.location.href=\"http://students.infoiasi.ro/~puco/lic\";
self.moveTo(0,0);
self.resizeTo(screen.availWidth,screen.availHeight);
window.focus();
// -->
</script>";
exit(); }

// functia de verificare a URL-ului de la care provine votul
function check_referer($site_url, $referer) {
   if (eregi("(^$site_url)", $referer))
        return(true);
   return(false);
}
// functia care creeaza o fereastra de
// avertizare in JavaScript
function err_pop($msg) {
   echo "
      <script type='text/javascript'>
      <!--
      window.alert('$msg');
      // -->
      </script>";
}
?>
```

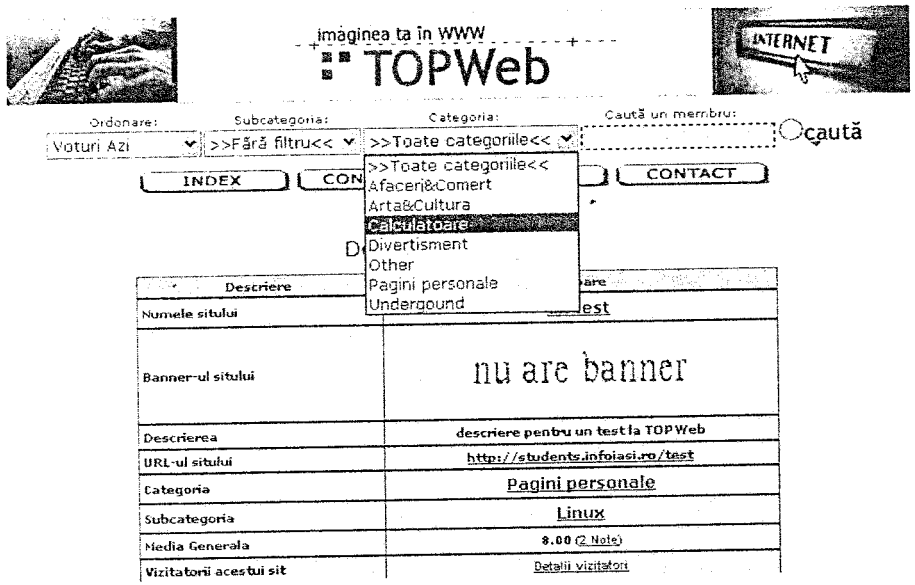
5. Contorizarea vizitatorilor

5.1. Introducere

În cadrul acestui subcapitol va fi prezentat sistemul de înregistrare unică a vizitatorilor, ca şi a numărului efectiv de vizitări a fiecărui sit membru.

Pentru ca un utilizator proprietar de sit să beneficieze de serviciile aplicaţiei *TopWeb*, trebuie ca paginile sitului său să aibă inclusă secvenţa de cod HTML primită de la această aplicaţie.

Un vizitator va fi identificat în mod unic de o înregistrare în tabela **voturi** care are 1 ca valoare a câmpului *visit*. Numărul de afişări va fi stocat în tabela **vizite**, în cadrul câmpului *Afisari*.



Vizualizarea informaţiilor privitoare la un sit membru al *TopWeb*

5.2. Algoritmul de contorizare a vizitatorilor

Vizitele pe siturile membre sunt înregistrate în momentul în care este afişată imaginea de vot. Astfel, în codul HTML pe care îl primeşte proprietarul sitului membru imaginea de vot este apelată astfel:

```
<img alt="Imaginea de vot"
src="http://students.infoiasi.ro/~puco/lic/visit.php?ID=6">
```

Fişierul *visit.php* contorizează afişarea respectivei pagini şi decide dacă vizitatorul provenind de pe un anumit sit este la primul vot pe ziua curentă pentru respectivul sit membru şi returnează imaginea de vot.

În consecinţă, vor fi contorizaţi vizitatorii şi numărul de afişări pentru un sit membru numai dacă în paginile sale este conţinută secvenţa de cod HTML furnizată de *TopWeb*.

Pentru a afişa imaginea de vot apelând fişierul *visit.php* în atributul *src* al elementului HTML `img` se foloseşte codul PHP următor:

```
// adresa butonului de vot
$adresa='./images/buttons/' . $vote_button;
header("Location: $adresa");
```

Variabila *$vote_button* este o variabilă a aplicaţiei, care va lua ca valoare un nume de fişier de tip **.gif**, **.png** sau **.jpg** (de exemplu, `buton_de_vot.gif`).

În continuare prezentăm sursa comentată a fişierului *visit.php*:

```
<?php
// fisierul visit.php este apelat cu parametrul ID
$id = $ID;
// este prezent parametrul ID

if ($ID) {
// verificam daca a fost vizitat astazi situl utilizatorului
// cu ID-ul $id de la acest IP
$query = mysql_query("SELECT * from voturi WHERE
   IP='$HTTP_SERVER_VARS[REMOTE_ADDR]'
   AND Data=CURDATE() and Visit='1' and ID='$id'")
   or die(mysql_error());
$found = mysql_affected_rows();

// a mai fost vizitata pagina astazi
// nu inseram o noua inregistrare in tabela voturi
if ($found > 0)
   { /* ... */ }

// NU a fost vizitata pagina $id de la acest IP
else {
// inserarea unei linii in tabela cu voturi
mysql_query("INSERT INTO voturi VALUES ($id,
   '$HTTP_SERVER_VARS[REMOTE_ADDR]',
   '$var_date', 1, 0)") or die(mysql_error()); }

// Urmeaza incrementarea numarului de afisari
// verificam daca a mai fost vizitat situl vizitatorului
```



```
// cu ID-ul pagina $id astazi de la acest IP
$query1= mysql_query("SELECT * from vizite
   WHERE IP='$HTTP_SERVER_VARS[REMOTE_ADDR]'
   AND Data='$var_date' and ID='$id'")
   or die(mysql_error());
$found1 = mysql_affected_rows();

// a mai fost vizitata - incrementam numarul de afisari
if ($found1 > 0) {
   mysql_query("UPDATE vizite SET Afisari=Afisari+1
   WHERE ID='$id' AND IP='$HTTP_SERVER_VARS[REMOTE_ADDR]'
   AND Data=CURDATE()") or die(mysql_error());
}
// nu a mai fost vizitata azi de la acest IP:
// inseram o noua linie si setam nr. de vizite la 1
else {
   mysql_query("INSERT INTO vizite
   VALUES ($ID, '$HTTP_SERVER_VARS[REMOTE_ADDR]',
   CURDATE(), 1)") or die(mysql_error());
}
}
// adresa butonului de vot
$adresa = './images/buttons/button_vot.gif';
// in pagina sitului utilizatorului
// este returnata imaginea de vot
// prin redirectionarea navigatorului
// via campul Location trimis in antetul HTTP
header("Location: $adresa ");

?>
```

6. Concluzii

Pe parcursul acestui capitol am prezentat conceptele şi algoritmii fundamentali ai unui sit care generează topuri de situri, sit denumit *TopWeb*, foarte asemănător prin funcţionalităţi unor aplicaţii de acest gen existente în peisajul actual al Web-ului românesc. Prezentarea este departe de a fi una completă, dar surprinde aspectele esenţiale ale unui astfel de sit. Cititorul interesat poate experimenta şi alte implementări ale părţilor de cod prezentate pe parcursul capitolului de faţă şi, desigur, poate dezvolta o versiune completă a aplicaţiei *TopWeb*.

Comentarii bibliografice

Pentru mai multe detalii despre funcţiile PHP şi MySQL folosite în scripturile prezentate vizitaţi adresele:

- **http://www.php.net/manual/en**
- **http://www.mysql.com/documentation**

A se vedea şi lista de referinţe prezentată în capitolul anterior şi, desigur, bibliografia generală de la finalul acestei cărţi.

Referitor la *cookie*-uri şi expresii regulate, cititorul poate găsi informaţii în cadrul cărţii *Programare Web în bash şi Perl*, concepută de S. Buraga, V. Tarhon-Onu şi Ş. Tanasă (vezi mai ales capitolele 2 şi 4).

CAPITOLUL 3

E-Shop IT – dezvoltarea unui magazin virtual

Ciprian Ginghină, snake@infoiasi.ro

Prezentul capitol expune un exemplu de dezvoltare a unei aplicaţii de comerţ electronic: un magazin virtual. Pornindu-se de la enunţarea principalelor obiective şi cerinţe ale unei astfel de aplicaţii, se continuă cu detalierea unei posibile soluţii software şi cu prezentarea succintă a soluţiilor alternative. În încheiere se oferă un scurt manual de utilizare a aplicaţiei *E-Shop IT*.

1. Obiectivele şi cerinţele unui magazin virtual

O dată cu răspândirea spaţiului WWW şi cu atingerea unei mase critice de utilizatori, atât firmele cu activitate comercială, cât şi analiştii şi specialiştii în marketing au ajuns la concluzia că merită investit efort în punerea la punct a unor tehnologii care să transforme utilizatorii Web în potenţiali clienţi.

Iniţial, siturile Web ale acestor firme conţineau doar informaţii de contact, oferte promoţionale sau chiar cataloage de produse sub forma unor pagini HTML statice. Lansarea unei comenzi de cumpărare a unor produse se putea face eventual via fax, telefon sau *e-mail*. Totul urma un curs firesc, doar că pentru a putea intra în posesia produselor clientul trebuia să plătească contravaloarea produselor comandate, iar banii să parcurgă drumul de la client la comerciant (vânzător), apoi de la acesta la producător.

Plata se realiza de regulă prin mecanismele clasice, către un cont deschis de comerciant la o bancă; presupunea deplasarea clientului la sediul unei instituţii cu profil bancar pentru a depune banii şi pentru a iniţia transferul acestora în contul comerciantului. În funcţie de politica adoptată de comerciant pentru a demara procesul de livrare a produselor către client acesta cerea, de cele mai multe ori, confirmarea efectuării plăţii prin fax. Fie că se folosea o reţea de distribuţie proprie, fie că se apela la un serviciu poştal specializat, această ultimă fază devenea cea mai lungă.

Ideea că într-o bună zi orice tranzacţie economică va fi condusă exclusiv prin Internet, eficientizându-se astfel foarte mult sistemul de tranzacţionare, i-a făcut pe

mulţi manageri să înceapă să se intereseze cât de departe pot merge cu iniţiativele lor de tip **e-business**. Au început, aşadar, să se intereseze asupra soluţiilor existente, capabile să suporte cu succes procesul de aprovizionare *on-line* al unei întreprinderi sau sistemul de vânzare şi cumpărare procesate prin Internet. Iar toate aceste soluţii trebuiau să fie caracterizate de o flexibilitate mare în dezvoltare şi adaptabilitate rapidă la cerinţele specifice clientului (în condiţiile unor preţuri accesibile chiar şi întreprinderilor mici şi mijlocii de la noi).

Din cele prezentate mai sus, se observă o strânsă legătură între aplicaţiile Web şi comerţul electronic. Una dintre formele de manifestare ale acestui domeniu sunt magazinele virtuale, cunoscute şi sub numele de **e-shop**.

E-Shop IT reprezintă un magazin virtual, destinat celor interesaţi de componente hardware, dar la fel de bine această aplicaţie îşi poate schimba domeniul de activitate datorită modului flexibil în care a fost concepută.

Ce trebuie să ştim despre un e-shop?

Când dorim să dezvoltăm un magazin virtual (*e-shop*) trebuie avuţi în vedere următorii paşi care trebuie urmaţi pentru a fi siguri că magazinul nostru va avea succesul scontat.

Să ne imaginăm, de exemplu, că deţinem o companie care se ocupă de telefonie fixă şi mobilă. Am dori să creştem vânzările acestei companii. Ce posibile soluţii avem la dispoziţie? Putem apela la mecanismele clasice de publicitate (de exemplu, spoturi radio sau TV) şi/sau putem amplasa reclame în zonele aglomerate în speranţa că vom atrage potenţialii clienţi.

O soluţie care în ultimii ani s-a dezvoltat foarte mult şi a devenit foarte la modă o reprezintă magazinele virtuale (*e-shop*). Oricine poate fi clientul acestor magazine virtuale. Este suficientă o conexiune la Internet şi un *browser* pentru navigarea *on-line* în cadrul magazinului.

În prezent, există o varietate de *e-shop*-uri, oferind clienţilor o multitudine de servicii. Dintre cele mai importante servicii putem aminti următoarele: coşul virtual de cumpărături, catalogul de produse cu serviciile specifice, posibilităţile de plată, autentificarea utilizatorilor, înscrierea la diferite liste de discuţii şi multe altele.

Orice programator poate realiza un magazin virtual. Nu avem nevoie decât de un editor de texte, un server Web (Apache, de exemplu), un limbaj de script (de exemplu, PHP) şi cunoaşterea fundamentelor HTML.

Atunci de ce mai apelăm la firmele de design Web şi nu dezvoltăm singuri un astfel de magazin? Pentru că, pentru a face faţă cu succes concurenţei şi celor mai pretenţioase cerinţe ale clienţilor, trebuie să recurgem la talentul şi experienţa profesioniştilor. Din cele descrise mai jos, vom vedea limpede că putem reuşi să implementăm un *e-shop* dacă avem unele cunoştinţe – nu foarte avansate – din domeniul tehnologiilor Web.

Care sunt principalele obiective ale unui e-shop?

Ca obiective principale ale magazinelor virtuale se doresc a fi creşterea vânzărilor, comunicarea permanentă cu clienţii, livrarea produselor într-un timp cât mai scurt, precum şi asigurarea unor modalităţi de plată cât mai sigure şi cât mai eficiente.

Cum putem realiza aceste obiective? În primul rând se impune o analiză atentă a categoriilor de produse care vor fi scoase la vânzare. Realizarea unor statistici periodice, actualizarea permanentă a listelor de produse oferite şi răspunsul prompt la cerinţele clienţilor pot face din magazinul virtual o afacere electronică de invidiat.

Care sunt principalele cerinţe ale unui e-shop?

Cerinţele magazinului virtual sunt de fapt cerinţele clienţilor. O problemă deosebit de importantă de care trebuie ţinut cont este migrarea de la o piaţă locală la una globală. Ca urmare a acestei migrări, categoriile de clienţi vor fi mult mai diversificate, dar la fel şi cerinţele lor.

Dintre cerinţele cele mai importante amintim cerinţele de securitate, cerinţele de design, cerinţele de implementare şi eficienţă, cerinţele de publicitate.

Dacă nu vom avea în vedere satisfacerea acestora, magazinul nostru nu va avea parte decât de un eşec răsunător.

Pentru mai multe detalii privitoare la *e-commerce* în particular şi la *e-business* în general se poate consulta capitolul 3 al cărţii *Proiectarea siturilor Web*, scrisă de Sabin Buraga.

2. Soluţia software

Soluţia software aleasă pentru dezvoltarea acestui magazin virtual este cea folosind PHP ca limbaj de programare, MySQL drept server de baze de date şi Apache ca server Web. Principalele avantaje oferite de această soluţie sunt următoarele: interacţiunea cu vizitatorul şi mentenenţa sitului. Ca soluţii alternative putem aminti: *ASP/ASP.NET + MS SQL Server + IIS* (*Internet Information Services*) – în special pe o platformă Windows – sau *JSP + MySQL + Tomcat/Apache* – dacă dorim să ne îndreptăm atenţia spre o soluţie Java (pentru aceasta din urmă se poate consulta lucrarea *Java de la 0 la expert*, de Ş. Tanasă, C. Olaru şi Ş. Andrei, apărută în anul 2003 la Editura Polirom).

Un sit cu conţinut dinamic permite interacţiunea cu utilizatorul. Acesta poate căuta, selecta şi chiar personaliza informaţia pe care o doreşte; sau poate contribui la îmbogăţirea cu conţinut a acestuia. Însă principalul avantaj al unui sit Web dinamic, rezidă în simplificarea procesului de mentenenţă (întreţinere).

Întreţinerea unui sit al cărui conţinut se modifică frecvent este unul din principalele aspecte de care trebuie ţinut cont. Iar acest lucru va fi valabil şi pentru aplicaţia de faţă. Rareori se întâmplă într-o situaţie reală ca în cadrul bugetului iniţial, alocat pentru dezvoltarea unui sit Web (catalog), să fie avute în vedere şi cheltuielile ulterioare pentru întreţinerea şi actualizarea conţinutului. Dacă tehnologia aleasă nu va fi suficient de simplă, dar totodată eficientă, pentru a fi stăpânită de cei care o vor utiliza efectiv, finalul nu poate fi decât un eşec.

3. Arhitectura aplicaţiei

3.1. Clasificarea în module şi interacţiunea dintre ele

În această secţiune, vom structura cerinţele aplicaţiei în patru module şi vom discuta despre modul în care comunică între ele. Vom încerca să evidenţiem problemele importante care pot apărea în timpul comunicării dintre module.

Aplicaţia *E-shop IT* a fost dezvoltată pornind de la următoarele module:

- autentificarea;
- catalogul de produse şi serviciile oferite;
- transferul bancar;
- administrarea aplicaţiei.

Autentificarea în cadrul aplicaţiei *E-shop IT* se referă la maniera de identificare a utilizatorului. Identificarea se realizează printr-un nume de utilizator şi o parolă, care au fost alese de utilizator în momentul completării formularului de înscriere. După înscriere şi autentificare, utilizatorul poate beneficia de toate serviciile oferite de aplicaţie, despre care vom discuta mai jos.

Ce probleme pot apărea? Introducerea unor date invalide în formularul de autentificare va produce un mesaj de avertizare din partea aplicaţiei. De exemplu, dacă am uitat numele de utilizator (cont – aşa-numitul *login name*) sau parola nu vom putea să ne autentificăm şi vom primi un mesaj prin care suntem informaţi despre incorectitudinea datelor introduse (*i.e.* un cont inexistent sau o parolă greşită).

Ce soluţie avem? Aplicaţia ne oferă o modalitate de a ne reaminti parola şi contul. Trebuie să completăm un formular, în care vom introduce adresa de *e-mail* utilizată în formularul de înscriere.

Cum este stocată parola? În momentul în care un nou client doreşte să-şi creeze un cont, valoarea câmpului parolă va fi criptată. Pentru criptare am folosit funcţia predefinită crypt() disponibilă în PHP. Această funcţie are doi parametri: şirul care se doreşte a fi criptat şi un al doilea parametru care este opţional, reprezentând o cheie de criptare şi specificând ce algoritm va fi folosit pentru criptare. Algoritmii cei mai utilizaţi sunt *DES* (*Data Encryption Standard*) sau *MD5* (*Message Digest Algorithm*) – a se consulta şi cartea *Securitatea sistemelor Linux* scrisă de D. Acostăchioaie.

Pentru a cripta folosind algoritmul *DES*, funcţia se va apela astfel: `crypt("sir de criptat", substr("sir de criptat", 0, 2))`. Se va utiliza o cheie formată din două caractere, extrasă chiar din şirul ce va fi criptat.

Pentru celălalt algoritm, *MD5*, funcţia se va apela astfel: `crypt("sir de criptat", "$1$sirdecri")`. Cheia folosită trebuie să conţină douăsprezece caractere şi să înceapă cu simbolurile „`$1$`".

Din materialele studiate despre aceşti doi algoritmi am ajuns la concluzia că pentru aplicaţia noastră vom utiliza algoritmul *MD5* ca fiind o soluţie fezabilă în ceea ce priveşte securitatea oferită. A se vedea şi secţiunea 3.3 a celui de-al doilea capitol al cărţii de faţă.

Să discutăm acum despre modul de reamintire a parolei. Atunci când clientul va selecta această opţiune, îi va apărea un formular în care va trebui să completeze anumite date folosite la înscriere (nume, prenume, adresa de *e-mail*, numele de cont). Pentru a putea folosi vechiul cont, va trebui să introducă o nouă parolă, care va fi criptată şi va fi stocată într-o bază de date. Deci nu este vorba despre o reamintire a parolei, pentru că acest lucru nu este posibil. Motivul este simplu. Toate parolele sunt generate folosind algoritmul *MD5*, plus o anumită cheie de criptare. Pentru reamintirea parolei, ar trebui să ştim parola clientului necriptată. Dar aplicaţia nu reţine şi parolele necriptate pentru că nu ar mai avea sens folosirea algoritmului de criptare.

Clientul va alege o nouă parolă, va completa celelalte date şi în cazul în care datele corespund cu cele stocate în baza de date (mai puţin parola), se va modifica vechea parolă pentru a permite accesul clientului la vechiul cont. Dacă utilizatorul nu-şi reaminteşte datele cerute în formular, singura soluţie va fi crearea unui nou cont.

Datele introduse în formular, inclusiv parola, nu vor putea fi interceptate de o terţă persoană deoarece se foloseşte protocolul *SSL* (*Secure Socket Layer*) pentru a transmite datele la server (via protocolul HTTPS). Orice încercare de interceptare a datelor va fi blocată de acest protocol.

Catalogul de produse şi servicii oferite reprezintă, pe scurt, un mecanism de vizualizare a produselor, facilitând selectarea lor şi efectuarea unor acţiuni specifice procesului de cumpărare a unor bunuri. Pentru a exemplifica ce am spus, considerăm că dorim să cumpărăm un monitor prin intermediul acestei aplicaţii. Alegem categoria „monitoare" din lista de categorii disponibile, selectăm unul dintre produsele acestei categorii prin folosirea butonului **add to cart** atunci când vizualizăm o descriere detaliată a produsului sau butonul **buy now** atunci când alegem direct produsul, fără a consulta detaliile tehnice. În continuare, avem următoarele servicii la dispoziţie: selectarea altor produse din aceeaşi categorie sau din categorii diferite, modificarea cantităţii fiecărui produs, ştergerea din lista de produse selectate pentru a fi cumpărate, a celor care nu ne mai interesează sau putem şterge toate produsele o singură dată folosind butonul **delete cart**.

Prin acest exemplu am ilustrat modul în care se creează un coş de cumpărături virtual şi ce operaţii putem efectua cu el.

Un serviciu foarte important, despre care vom discuta în modulul următor, este acela de a realiza transferul bancar.

Precizăm faptul că toate serviciile oferite cu excepţia ultimului sunt oferite fără autentificare. Doar atunci când dorim să plătim produsele alese trebuie să ne autentificăm, dacă nu am realizat acest lucru în prealabil.

Transferul bancar este un alt modul deosebit de important în cadrul unui magazin virtual. Ce înseamnă acest transfer? La fel cum în mod obişnuit când cumpărăm ceva dintr-un magazin trebuie să plătim, similar se întâmplă şi în cadrul unui magazin virtual. Modalităţile de plată sunt diverse, dar noi am optat pentru plata prin intermediul unui card bancar. În momentul selectării butonului **send the order**, clientul va completa un câmp al unui formular care va conţine seria cardului bancar. Alături de această informaţie mai sunt afişate în formular adresa serverului băncii, unde se va conecta clientul precum şi totalul sumei ce trebuie achitată. Dacă datele introduse

sunt validate, atunci mai este un singur pas pentru a încheia tranzacţia. Acest pas constă în utilizarea unui formular, în care vom răspunde la o întrebare despre succesul tranzacţiei până în acel moment. De exemplu, dacă toate validările au fost efectuate cu succes, răspunsul la întrebare va fi afişat automat. Clientul trebuie doar să utilizeze butonul de tip *submit* pentru a încheia tranzacţia.

O problemă importantă o reprezintă securitatea datelor transmise. De acest lucru se ocupă, de asemenea, protocolul SSL – tranzacţiile bancare trebuie desigur securizate.

Administrarea aplicaţiei este un modul de eficienţa căruia depinde succesul oricărui magazin electronic. Actualizările produselor fiind frecvente, este necesar ca acest modul să facă faţă cu succes unui număr (relativ) mare de actualizări zilnice.

Administrarea a fost dezvoltată într-un mod dinamic, la fel ca şi celelalte module. Acest modul nu este destinat utilizatorilor obişnuiţi, ci doar administratorilor care se ocupă de actualizarea produselor. Serviciile oferite de acest modul sunt următoarele:

- adăugarea de noi produse şi categorii;
- modificarea produselor din categoriile existente;
- ştergerea produselor din categoriile existente.

Adăugarea a fost concepută pornind de la ideea că nu toate produsele au acelaşi număr de caracteristici, fiind posibil ca produse din aceeaşi categorie să difere prin câteva detalii. Iniţial am dorit să organizăm înregistrările în mai multe tabele, folosind câte o tabelă pentru fiecare categorie. O dată creată o tabelă, ea are o anumită structură. Ce se întâmplă dacă apare o înregistrare care are mai multe câmpuri decât conţine tabela? Ce putem face în această situaţie? O soluţie ar fi modificarea structurii tabelei, ceea ce va genera o serie de probleme, atunci când numărul de produse adăugate este foarte mare şi descrierea produselor nu se potriveşte cu structura tabelei.

O altă soluţie, mai bună decât cea precedentă, este folosirea unei tabele a cărei structură să conţină caracteristicile comune tuturor produsele, încă un câmp de o dimensiune suficient de mare care să conţină o descriere particulară, corespunzătoare fiecărui produs. Câmpul folosit pentru a reţine descrierea complexă a produsului este de tip *BLOB* (*Binary Large OBject*) şi poate stoca maxim 2^{16}-1 octeţi. Informaţiile stocate în acest câmp vor putea fi procesate cu uşurinţă şi afişate după dorinţa fiecăruia.

După cum am spus şi mai sus, produsele pot fi stocate în tabele diferite, dar pot fi stocate şi în aceeaşi tabelă. În aplicaţia noastră am utilizat, pentru afişarea dinamică a categoriilor şi produselor trei tabele: o tabelă în care avem toate produsele identificate unic prin *id*-ul lor, a doua tabelă conţine toate caracteristicile categoriilor existente şi a treia stochează codul categoriei şi denumirea explicită.

Modificarea produselor se realizează în mare parte la fel ca şi adăugarea. Un plus îl constituie modul în care se realizează modificarea descrierii complexe a unui produ- Putem modifica atât denumirea caracteristicii, cât şi valoarea ei. Toate operaţiile se realizează asupra unui şir, care în final va deveni valoarea câmpului de descriere complexă a produsului. Spre a exemplifica, presupunem că avem perechea (NUME, E-shop IT). Avem posibilitatea să o transformăm în (NAME, E-shop IT). După cum se observă, este un punct de plecare pentru a traduce descrierea detaliată a unui produs în diferite limbi, asigurându-se astfel internaţionalizarea aplicaţiei.

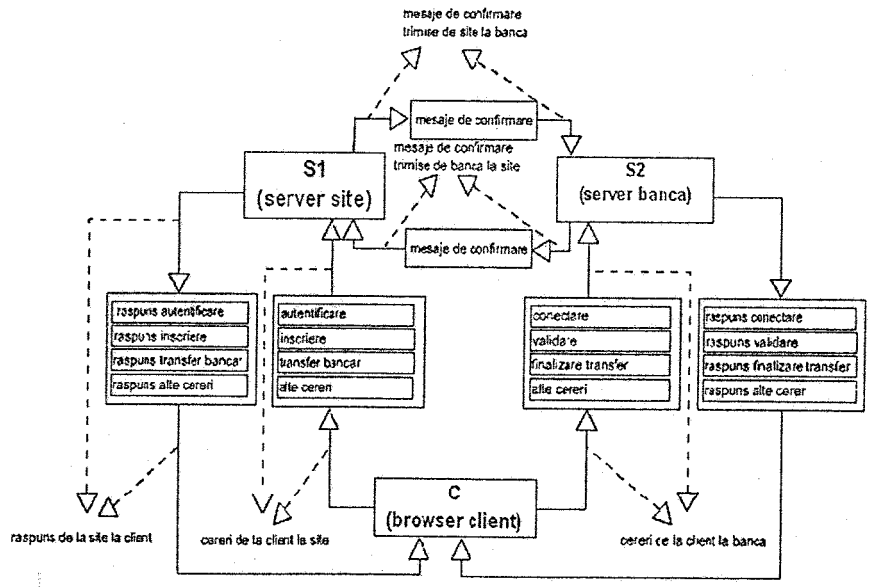
Ştergerea unei înregistrări se realizează prin selectarea categoriei de produse dorite şi folosirea butonului **delete**. Atunci când categoria conţine numai o singură înregistrare se va şterge şi categoria respectivă.

În continuare, pentru a putea exemplifica modul în care interacţionează modulele prezentate mai sus vom considera trei entităţi: serverul care găzduieşte aplicaţia **E-Shop IT** (S1), serverul băncii folosit pentru plata produselor (S2) şi *browser*-ul (navigatorul) clientului (C).

Clientul deschide prin intermediul *browser*-ului o conexiune la **S1**. Din acest moment **C** trimite cereri la **S1**. **S1** va returna răspunsul cererilor solicitate de **C**.

Ce fel de cereri poate transmite clientul?

- *cerere de autentificare*: modulul de autentificare primeşte datele din formularul existent în *browser*-ul clientului atunci când acesta va apăsa butonul de **login**. Datele primite vor fi validate şi **S1** va returna răspunsul cererii: un mesaj de eroare dacă datele sunt invalide sau o nouă pagină cu opţiunile clientului;
- *cerere de înscriere*: o componentă a modulului de autentificare este şi înscrierea de noi clienţi. Datele primite din formular vor fi validate, după care **S1** va returna răspunsul cererii: un mesaj de confirmare a noului cont sau mesaje de eroare pentru câmpurile formularului completate incorect;
- *cerere de utilizare a catalogului de produse*: clientul solicită anumite servicii, iar **S1** răspunde la aceste cereri de servicii;
- *cerere de transfer bancar*: această cerere este solicitată de client, **S1** nu o poate realiza pentru că de acest lucru se ocupă **S2**;



Schema interacţiunilor realizate în cadrul aplicaţiei *E-Shop IT*

În acest moment, după ce a trimis cereri către S1 şi a primit anumite răspunsuri, clientul va trimite cereri către S2.
Ce fel de cereri poate trimite clientul?

- *cerere de transfer bancar*: clientul completează un formular, datele vor ajunge la serverul băncii printr-o conexiune securizată (HTTPS). Vor fi verificate şi S2 va trimite un răspuns la client: dacă seria cărţii de credit este validă şi contul bancar este solvabil atunci clientul mai poate trimite o cerere la S2, cerere de încheiere a transferului bancar.

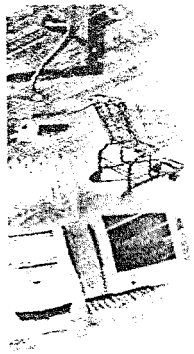
Un alt lucru deosebit de important pentru finalizarea tranzacţiei este şi comunicarea dintre S1 şi S2.

Nu este suficient ca datele trimise de client să fie valide pentru încheierea tranzacţiei. S1 trebuie să primească de la S2 un mesaj de confirmare a transferului bancar din contul clientului în contul sitului. Iar S1 trebuie să trimită mesaj la S2 cum că a primit mesajul de confirmare.

Ce se întâmplă în cadrul tranzacţiei bancare, dacă apare o problemă de comunicare între aceste entităţi? Vom avea următoarele scenarii:

- dacă C nu poate comunica cu S2, atunci clientul va fi nevoit să încerce conectarea la serverul băncii mai târziu;
- dacă C este capabil să comunice cu S2, dar S2 nu comunică cu S1 atunci S2 va păstra datele tranzacţiei până la momentul respectiv, pentru a putea relua mai târziu tranzacţia;
- la fel se întâmplă şi atunci când S2 comunică cu S1, dar S1 nu mai poate să comunice cu S2.

3.2. Autentificarea utilizatorului

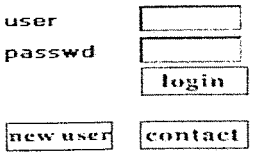


După cum am văzut în prima secţiune a acestui capitol, prin autentificare se înţelege modalitatea de recunoaştere a unui anumit client. Ne referim la identificarea numelui şi a datelor personale completate de client în formularul de înscriere. Cum realizăm acest proces de autentificare?

Vom prezenta şi explica cele mai importante secvenţe de cod PHP care realizează autentificarea.

După cum am menţionat, pentru autentificare trebuie date un nume de utilizator şi o parolă corespunzătoare acestui nume. Butonul **login** va transmite valorile din formular, prin intermediul serverului Web (Apache), la modulul de autentificare, care va prelucra datele şi va trimite clientului un răspuns la această cerere.

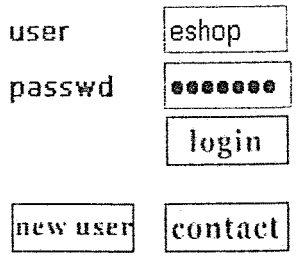
Funcţia *afiseaza_index( )* va genera formularul descris mai sus – codul ei este următorul:



Interfaţa de autentificare a unui utilizator


```
function afiseaza_index()
{
  global $cont;
  echo "<center>";
  echo "<table width='93%'  bgcolor='white' valign='top'";
  echo "<tr><td>";
  echo "<img name='buton1' src='gif/banere/shop.jpg' ";
  echo "onmouseover=\"
   movepic('buton1','gif/banere/shop1.jpg')\" ";
  echo "onmouseout=\"
   movepic('buton1','gif/banere/shop.jpg')\">";
  echo "</td></tr>";
  echo "</table></center>";

  // cream un formular pentru cont si parola
  echo "<form method='POST' action='script/login.php'>";
  // prima linie cont
  echo "<center>";
  echo "<table witdh='93%' border='0' cellspacing='0'
      cellpading='15'   valign='top' style='border:none'>";
  echo "<tr>";
  echo "<td border='1' width='60%' height='25%' align='left'
    bordercolor='#AEAEFF' > ";
    echo "<font class='text_cont_parola'>user</font>";
    echo "</td>";
    echo "<td border='1' width='40%' height='35%'
    align='left' bordercolor='#AEAEFF'>";
    echo "<input type='text' name='cont' maxlength='15'
    size='7'>";
    echo "</td>";
  echo "</tr>";
  // a doua linie parola
  echo "<tr>";
  echo "<td border='1' width='60%' height='25%'
    align='left' > ";
    echo "<b><font class='text_cont_parola'>passwd</font>
    </b>";
    echo "</td>";
    echo "<td border='0'  width='40%'height='25%'
    align='left'>";
    echo "<input type='password' name='parola' maxlength='15'
    size='7'>";
    echo "</td>";
  echo "</tr>";
```

```
// a treia linie login
echo "<tr>";
  echo "<td border='1' width='60%' height='25%'
  align='left'>";
  echo "</td>";
  echo "<td border='1' width='40%' height='25%'
  align='right' >";
  echo "<input type='image'  src='gif/butoane/login.jpg'
  value='login'  size='7'>";
  echo "</td>";
echo "</tr> ";
echo "<tr>";
  echo "<td  border='1' width='60%' height='25%'
  align='left'>  ";
  echo "<br><a  href='user/new_user.php'";
  echo onmouseover=\" movepic('buton2','gif/butoane/
  new_user1.jpg')\"";
  echo "onmouseout=\" movepic('buton2','gif/butoane/
  new_user.jpg')\">";
  echo "<img name='buton2' src='gif/butoane/new_user.jpg'
  border='0'>";
  echo "</a>";
  echo "</td>";
  echo "<td  border='1' width='40%' height='25%'
  align='left'>";
  echo "<br><a  href='mailto:snake@infoisi.ro' title=
  'send a message to admin'";
  echo "onmouseover=\" movepic('buton','gif/butoane/
  contact1.jpg')\"";
  echo "onmouseout=\" movepic('buton','gif/butoane/
  contact.jpg')\">";
  echo "<img name='buton' src='gif/butoane/contact.jpg'
  border='0'>";
```



Autentificarea unui utilizator


```
    echo "</a>";
    echo "</td>";
    echo "</tr>";
  echo "</table>";
  echo "</center>";
  echo "<input type='hidden'  name='message' value=
    '$message'>";
  echo "</form>";
}
```

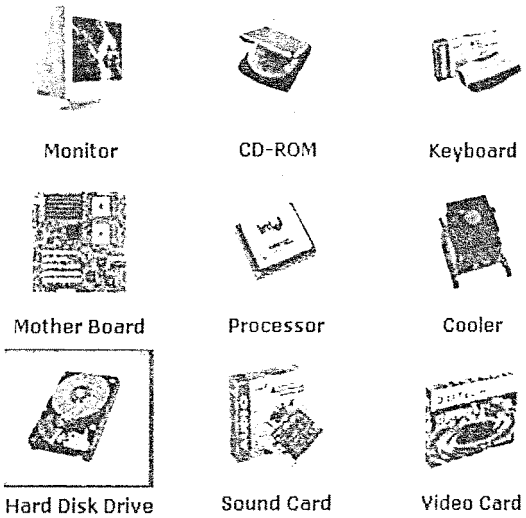
Datele de intrare vor fi procesate de scriptul *login.php* care face parte din modulul de autentificare. Numele de utilizator şi parola vor fi transmise, prin intermediul acestui script, funcţiei *verifica_user_inscris()* care va verifica dacă aceste informaţii există în tabela *useri*. Prezentăm mai jos codul ei.

```
function verifica_user_inscris($cont,$parola)
{
    // conectare la serverul de baze de date
    $server = mysql_connect("localhost","snake","snake");
    $connection = mysql_select_db("site",$s);

    if($connection)
    {
       // daca ne-am conectat atunci selectam
       // utilizatorii din tabela useri
      $result = mysql_query("select cont,parola from useri");
       while($line = mysql_fetch_array($result))
       {
       if($cont==$line["cont"] && $parola==$line["parola"])
       {
              return 0;
       }
    }
    return 1;
    mysql_close($server);
    }
    else
    {
       echo " Connection failed ";
       return 0;
    }
}
```

Dacă datele introduse în formular sunt valide, atunci utilizatorul se va autentifica cu succes şi va primi ca rezultat un mesaj de întâmpinare din partea aplicaţiei. Dacă datele trimise sunt invalide, atunci va fi afişat un mesaj de eroare.

3.3. Afişarea dinamică a categoriilor

Afişarea categoriilor disponibile este un serviciu oferit de catalogul de produse, specific oricărei aplicaţii de tip *e-shop*. Folosirea unei baze de date, care să conţină categoriile disponibile, conferă un caracter dinamic aplicaţiei.



Afişarea dinamică a categoriilor

Pentru actualizarea rapidă a informaţiilor conţinute în tabelele unei baze de date, avem nevoie de un modul de administrare foarte eficient, aşa cum am văzut mai sus când am vorbit despre modulul de administrare. De eficienţa acestui modul depinde modul de afişare a categoriilor şi a produselor specifice fiecărei categorii.

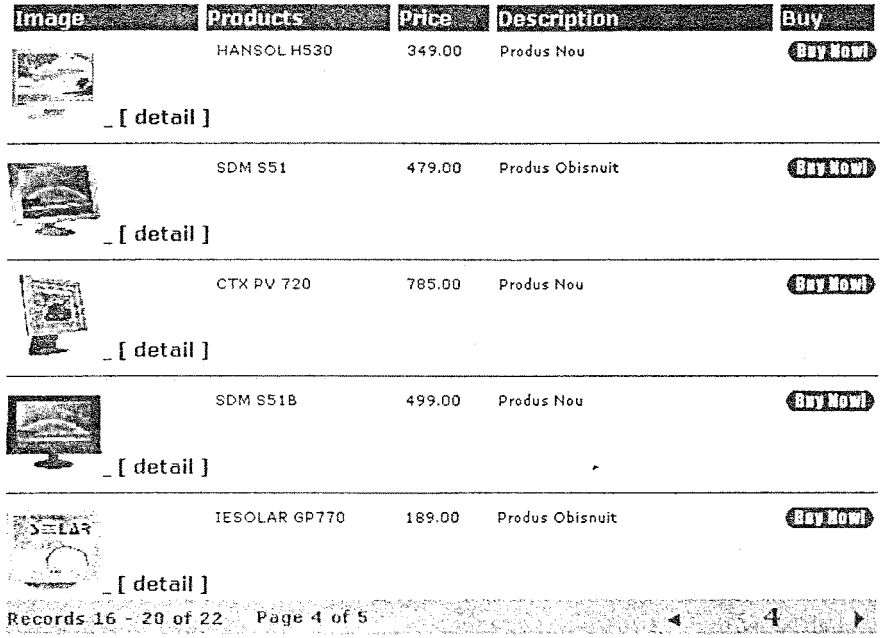
3.4. Afişarea dinamică a produselor

După ce am afişat categoriile, următorul pas este vizualizarea produselor din fiecare categorie. Maniera de afişare a înregistrărilor este asemănătoare cu cea folosită de motorul de căutare **Google** sau de situri Web cum ar fi: **www.rentacoder.com**, **www.trafic.ro**, **www.statistici.ro** etc.

Utilizatorul poate selecta numărul de produse afişate pe o singură pagină, precum şi numărul de legături spre paginile următoare. Figura de mai jos reprezintă un mod de vizualizare cu 5 înregistrări pe pagină şi 2 legături, cu un total de 5 pagini de informaţie.

Se observă divizarea modului de afişare a conţinutului pe coloane, fiecare coloană reprezentând valoarea unui atribut al înregistrărilor. Prima coloană conţine imaginea

produsului precum şi un text [**detail**]. Atât imaginea cât şi textul conduc prin intermediul unei legături spre o funcţie care afişează o descriere detaliată a produsului. Următoarele coloane conţin numele, preţul, tipul produsului şi un buton pentru a adăuga produsul în coşul de cumpărături.



Afişarea produselor din categoria *Monitoare*

În partea de jos, sunt disponibile informaţii despre numărul înregistrărilor afişate, numărul paginii curente, legături spre paginile următoare sau precedente. Funcţia *display()* se ocupă de generarea acestei afişări. Codul sursă este mai amplu şi din acest motiv vom prezenta doar aspectele mai importante.

```
function display($st, $ne, $start_page, $nr_page, $cod_produs)
{
global $start;
global $finish;
global $value;    //nr. de inreg pe pagina
global $_st;      //salveaza numarul de inceput pentru
                  //afisarea inregistrarilor
global $_fi;      //salveaza nr. pentru afisarea inregistrarilor
global $_st_p;    //salvarea numarului de inceput de pagina
global $_nr_p;    //salvarea numarului de sfarsit de pagina
```

```
$_st = $st;        //initializam cu nr. inreg de start;
$_fi = $ne;        //initializam cu nr. inreg de sfarsit;
$_st_p = $start_page;    //initializam cu nr. paginii de start
$_nr_p = $nr_page;       //initializam cu nr. paginii de final

// transmitem parametrul pentru o noua apelare
$tabela = $table;
// alegem tipul de produs
if($cod_pro != "all")
{
    $connection = mysql_connect("localhost","snake","snake");
    $server = mysql_select_db("produse",$s);
    if($server)
    {
    $result = mysql_query("select * from categorii where
    cod_produs='$cod_produs'");
    $director = mysql_result($result,0,"tip_produs");
    }
    else
        echo "Connection failed";
    mysql_close($s);
}
antet();
?>
<table width="100%" align="top">
<tr>
<td>
<!-- AFISARE PRODUSE -->
<?
    include("../db/db.php");
    $connection = @ConnectToDb
                    ($dbServer, $dbUser, $dbPass, $dbName);
    $result = mysql_query("select * from produse where
    cod_produs='$cod_produs'");
?>
<!--ANTETUL DIN TABEL INCLUS IN COLOANA TABELULUI SECUNDAR-->
<div align="center">
<table width="90%" cellspacing="5" cellpading="5" >
<tr>
<td  width="20%" bgcolor="#075ca8" > <font class='text_antet'>
    <b>Image</b></font> </td>
<td  width="20%" bgcolor="#075ca8"> <font class='text_antet'>
    <b>Products  </b></font> </td>
```



```
<td  width="10%" bgcolor="#075ca8"> <font class='text_antet'>
    <b>Price      </b></font> </td>
<td  width="30%" bgcolor="#075ca8">  <font class='text_antet'>
    <b>Description </b></font></td>
<td  width="10%" bgcolor="#075ca8">  <font class='text_antet'>
    <b>Buy          </b></font></td>
</tr>
</table>
</div>
<!--PRODUSELE DIN BAZA DE DATE
    DUPA CE IN PREALABIL NE-AM CONECTAT LA EA-->
<?
//daca numarul de inegistrari pe pagina este mai mare decat
//cel existent atunci value = nr_max_inreg si ne = nr_max
if($value > mysql_num_rows($result))
{
    $value = mysql_num_rows($result);
    $ne = mysql_num_rows($result);
}
 echo "<div align=\"center\">    ";
 echo "<table width=\"90%\" border=\"0\" cellspacing=\"0\"
    cellpading=\"2\"
 valign=\"top\">";

//procesarea inregistrarilor din baza de date
for($i = $st; $i < $ne; $i++)
{
$poza_mica = "../gif/".$director."/mic/".mysql_result($result,
    $i,"cod_produs")."_".mysql_result($result,$i,"id").".".
    mysql_result($result,$i,"ext");
$poza_mare = "../gif/".$director."/mare/".mysql_result
    ($result,$i,"cod_produs")."_". mysql_result($result,$i,
    "id").".".mysql_result($result,$i,"ext");
//afisam dinamic pentru fiecare inregistrare:
//imagine, denumire, descriere, pret, submit
    if(!is_file($poza_mica) || !is_file($poza_mare))
    {
    $poza_mica = "../gif/butoane/noimage.gif";
    $poza_mare = "../gif/butoane/noimage1.gif";
}
?>
 <tr>
    <!-- apelam functia pt afisarea detaliata -->
    <td width="22%" height="50" valign="top">
```

```
<?
    //afisam link pt detalii numai pt cantitate > 0
    if(mysql_result($result, $i, "buc") > 0)
    {
?>
<a href="products.php?action=detail&image=<?=$poza_mare;?>
    &cod_pro=<?=$cod_produs;?>& id=<?=mysql_result($result,
    $i,"id");?>&price=<?=mysql_result($result,$i,"pret");?>
    &descriere=<?=mysql_result($result,$i,"desc_c");?>">
    <img src="<? echo $poza_mica; ?>" width="65" height="55"
    border='0'>
 </a>
<a  class="categorii"
    href="products.php?action=detail&image=<?=$poza_mare;?>
    &cod_pro=<?=$cod_produs;?>
    &id=<?=mysql_result($result,$i,"id");?>
    &price=<?=mysql_result($result,$i,"pret");?>
    &descriere=<?=mysql_result($result,$i,"desc_c");?>">
     [ detail ]
 </a>
<?
}
    else
    echo "<img src=$poza_mica width=\"65\" height=\"55\"
    border='0' valign='top'>";
?>
        </td>
<td width="20%" height="25" valign="top" align="left">
<font class='text_mic'> <?php echo mysql_result($result,$i,
    "denumire"); ?> </font> </td>
<td width="10%" height="25" valign="top" align="left">
<font class='text_mic'><?  Echo mysql_result($result,$i,
    "pret"); ?> </font>      </td>
<td width="30%" height="25" valign="top" align="left">
<font class='text_mic'> <?php echo mysql_result($result,$i,
    "desc_s"); ?> </font> </td>
 <td width="10%" height="25" valign="top" align="left"> <font
    class='text_mic'>
<?
if(mysql_result($result, $i, "buc") > 0)
{
    //afisam un buton pentru a adauga un produs in cos
?>
```



```
<form action="../cart/cart.php?action=add_item &id=<? echo
    mysql_result($result,$i,"id");?>&qty=1&cod_pro=<?=
    $cod_produs;?>" method="POST">
<input type='image' src='../gif/butoane/buy.gif'
        name="submit" value="add  item" >
</form>

</td>
<?
    }
    else
        // daca avem cantitatea = 0 afisam un mesaj
        echo "<font color='red'>unavailable</font>";
?>
</tr>
<?
if($i != $ne-1)
{
?>
<tr>
<td width="100%" colspan="5">
   <hr size="1" color="red" noshade="noshade" />
</td>
</tr>
   <?
     }//endif
}//end for

//...
}
```

3.5. Coşul de cumpărături

Coşul de cumpărături şi serviciile oferite de acesta le vom explica ţinând cont de imaginea de mai jos care reprezintă un coş de cumpărături conţinând cinci produse. Fiecare produs selectat spre a fi cumpărat este afişat pe coloane conţinând cantitatea, denumirea şi preţul produsului şi opţiunea de ştergere **Remove**.

Cantitatea poate fi modificată, numărul de produse maxim ales fiind egal cu cantitatea maximă din tabela care conţine produsele. În dreapta jos a imaginii este afişat costul total al produselor şi opţiunea de transfer bancar despre care vom discuta în amănunt mai târziu. În coş mai pot fi adăugate produse folosind legătura **Keep Shopping**.

Funcţiile implementate pentru corecta funcţionare a coşului sunt următoarele: *AddItem(), UpdateItem(), RemoveItem(), ShowCart()*.

Funcţia *AddItem()* primeşte ca parametru *id*-ul produsului, cantitatea şi codul categoriei din care face parte produsul.

Cum se realizează de fapt adăugarea unui produs în coşul virtual? Mecanismul este simplu. Se transmit acestei funcţii parametrii amintiţi mai sus, apoi se inserează în tabela **cart** o nouă înregistrare, care conţine *id*-ul produsului, cantitatea, un identificator corespunzător sesiunii clientului şi numele de utilizator.

Avem două posibilităţi de a adăuga un nou produs:

- folosind butonul **buy now** din pagina corespunzătoare afişării produselor;
- a doua posibilitate este utilizarea legăturii **add to cart**, din descrierea detaliată a unui produs.

Qty	Products	Price	Option
1	531B	$115.00	Remove
1	CTX VL700F	$179.00	Remove
1	Hansol 710D	$171.00	Remove
1	Multiscan E450	$519.00	Remove
1	Multiscan TE988	$189.00	Remove

Keep Shoping Total: 1,173.00 $

send the order

Coşul de cumpărături

Codul funcţiei este dat mai jos:

```
function AddItem($itemId, $qty, $cod_produs)
{
    global $sesiune;
      // variabila globala reprezentand sesiunea curenta

    $file = make_file($sesiune,"cont");
    $citeste = read_file($file);
    $cont = return_value($citeste,"cont");

    $connection = mysql_connect("localhost","snake","snake");
    $server = mysql_select_db("produse",$s);
    (if $server)
    {
    $result = mysql_query(" select count(*) from cart where
    cookieId = '" . GetCartId() ."' and itemId = $itemId and
```



```
    cont       ='$cont' ");
       $row = @mysql_fetch_row($result);
    $numRows = $row[0];
    if($numRows == 0)
    {
   mysql_query("insert into cart(cookieId, itemId, qty, cont)
    values('" . GetCartId() . "' , $itemId, $qty, '$cont')");
    }
    else
    {
       UpdateItem($itemId, $qty, $cod_produs); // actualizam
    }
}
else
    echo " Connection failed ";
}
```

Funcţia *UpdateItem()* modifică numărul de produse selectate spre a fi cumpărate. Iniţial, când adăugăm un produs, cantitatea este egală cu unu, apoi cu ajutorul acestei funcţii putem selecta un număr mai mare de produse, în limita celor disponibile în tabela de produse. Prezentăm mai jos codul ei:

```
function UpdateItem($itemId,$qty,$cod_produs)
{
    global $sesiune;

    $file = make_file($sesiune,"cont");
    $citeste = read_file($file);
    $cont = return_value($citeste,"cont");

    $connection = mysql_connect("localhost","snake","snake");
    $server = mysql_select_db("produse",$s);
    if ($server)
    mysql_query("update cart set qty=$qty where cookieId = '"
    . GetCartId() .
    "' and itemId = $itemId and cont='$cont' ");
   else
    echo" Connection failed ";
}
```

Funcţia *RemoveItem()* şterge un produs din coşul virtual. Primeşte ca parametri *id*-ul produsului şi codul categoriei din care face parte produsul.

Codul-sursă al acestei funcţii este următorul:

```
function RemoveItem($itemId,$cod_produs)
{
```

```
    global $sesiune;

    $file = make_file($sesiune, "cont");
    $citeste = read_file($file);
    $cont = return_value($citeste, "cont");

    $connection = mysql_connect("localhost","snake","snake");
    $server = mysql_select_db("produse", $s);
    (if $server)
    mysql_query("delete from cart where cookieId = '" .
   GetCartId() ."'  and itemId = $itemId and cont ='$cont'");
    else
        echo" Connection failed ";
}
```

Ultima funcţie care se ocupă de afişarea coşului virtual la un moment dat este *ShowCart()*:

```
function ShowCart($cod_produs)
{
   global $_st;
   global $_fi;
   global $_st_p;
   global $_nr_p;
   global $sesiune;

   antet();
   /****** citim valoarea contului din fisier *******/
   $file = make_file($sesiune, "cont");
   $citeste = read_file($file);
   $cont = return_value($citeste, "cont");

   $connection = mysql_connect("localhost","snake","snake");
   $server = mysql_select_db("produse", $s);
   (if $server)
   {
    $result = mysql_query(" select * from cart inner join
    produse on cart.itemId = produse.id
    where cart.cookieId = '" . GetCartId() . "' and cont =
    '$cont' order by produse.denumire asc ");
?>
   <div align="center">
   <table width="80%" cellspacing="5" cellpading="5" >
    <tr>
    <td width="15%" bgcolor="#075ca8">  <font
```



```
    class='text_antet'><b>Qty      </b> </font>  </td>
    <td width="55%" bgcolor="#075ca8">
    <font class='text_antet'><b>Products</b> </font>  </td>
    <td width="20%" bgcolor="#075ca8">  <font
    class='text_antet'><b>Price   </b> </font>  </td>
}
<td width="10%" bgcolor="#075ca8">  <font
    class='text_antet'><b>Option  </b> </font>  </td>
   </tr>
   </table>
   </div>
   <div align="center">
   <table width="80%" cellpading="5" cellspacig="5">
<?
   while($row = mysql_fetch_array($result))
   {
    //calcul cost total
    $totalCost += ($row["qty"] * $row["pret"]);
?>
   <tr>
     <td width="16%" height="25"> <font face="verdana"
    size="1" color="black">
<select style="width:40px"  size="1" name="<?php echo
    $row["itemId"];  ?>" " onChange = "UpdateQty(this)">
<?
    for($i = 1; $i <= $row["buc"]; $i++)
    {
        echo "<option value='$cod_produs'";
        if($row["qty"] == $i)
        {
                echo " SELECTED ";
        }
       echo ">" . $i . "</option>";
    }
?>
</select>
     </font>
     </td>
     <td width="55%" height="25"> <font class='text_mic'>
   <?php  echo $row["denumire"]; ?> </font> </td>
 <td width="20%" height="25"> <font class='text_mic'>
 $<?php  echo number_format($row["pret"], 2, ".", ","); ?>
</font></td>
     <td width="10%" height="25"> <font face="verdana"
```

```
    size="2" color="black">
     <a class='remove' href="../cart/cart.php?action=
    remove_item &id=<?php echo $row["itemId"];?> &cod_pro=<?=
    $cod_produs;?>">
    Remove</font>
     </a>
     </td>
     </tr>
<?
?>
<?php
    }//inchidem while
?>
   <tr>
      <td width="100%" colspan="4">
          <hr size="1" color="red" noshade>
      </td>
   </tr>
   </table>
   </div>

   <div align="center">
   <table width="80%">
   <tr>
<?php
    if($cod_produs!="" && $_st!="" && $_fi!=""
       && $_st_p!="" && $_nr_p!="")
    {
?>
      <td width="70%" colspan="1" bgcolor="#8ebbe6">
<a class="keep_add" href="../cart/products.php?action=disp&st=
    <?php echo $_st; ?>
&next=<?php echo $_fi; ?>&start_p=<?php echo $_st_p;?>
&nr_p=<?php echo $_nr_p; ?>&cod_pro=<?=$cod_produs;?>">
    Keep Shoping </a>
  </td>
<?
    }
    else{
?>
      <td width="70%"  colspan="1" bgcolor="#8ebbe6">
  <a class="keep_add" href="../cart/product.php">Keep Shoping
    </a></td>
<?
```



```
}
?>
        <td width="30%"  bgcolor="#8ebbe6" align="right">
        <font class="text_total">  <b> Total:
<?php echo number_format($totalCost, 2, ".", ","); ?> $</b>
        </font>
        </td>
   </tr>
   <tr>
      <td width='30%'> </td>
      <td width="70%"  align='right'>
      <font color="#8ebbe6" size="2"><b>
  <form action="../cart/cart.php?action=show_form&summ=
        <?=$totalCost;?>&user=<?=$cont;?>" method="POST">
      <input type="submit" name="submit"
             value="send the order">
      </form>
      </b></font>
      </td>
   </tr>
   </table>
   </div>
<script language="JavaScript">
function UpdateQty(item)
{
    itemId = item.name;
    newQty = item.options[item.selectedIndex].text;
    newcod = item.options[item.selectedIndex].value;
    document.location.href = '../cart/cart.php?action=
    update_item&id='
    + itemId + '&qty=' + newQty + '&cod_pro=' + newcod;
}
</script>
<?
}
else
        echo" Connection failed ";
} //inchidem functia
?>
```

3.6. Serviciul de plată prin transfer bancar

Transferul bancar este un serviciu oferit de aplicaţie, atunci când clientul doreşte să plătească produsele selectate.

Cum a fost dezvoltat acest serviciu? Am pornit de la ideea conform căreia aplicaţia să nu aibă acces la seria de card a clientului. Seria de card va fi introdusă într-un formular, care va transmite datele printr-o conexiune securizată (HTTPS) la serverul băncii care a emis cardul respectiv. Am ales această metodă pentru a spori securitatea transferului.

Vom prezenta mai jos modul de funcţionare a acestui transfer bancar. Funcţia *validare_transfer()* se ocupă de validarea acestui serviciu. Parametrii primiţi de această funcţie sunt: numele de cont şi seria de card corespunzătoare aplicaţiei, numele de cont şi seria de card corespunzătoare clientului şi suma care va fi transferată dintr-un cont în altul.

Cum validăm aceste date, după ce au ajuns la serverul băncii?

În primul rând se verifică existenţa cardurilor bancare. De acest lucru se ocupă funcţia *serie_card()*, care returnează 1 dacă se confirmă existenţa cardului şi 0 altfel. Dacă seriile de card există, următorul pas este să verificăm dacă clientul este solvabil. Funcţia *verifica_suma()* returnează 1 dacă clientul este solvabil şi 0 altfel.

Dacă clientul este solvabil, putem începe transferul. Mai întâi trebuie să trimitem nişte mesaje la serverul aplicaţiei pentru a confirma succesul transferului.

Ce fel de mesaje trimitem? Mesajele sunt de forma **yes** sau **no**. Mesajul „yes" pentru o acţiune încheiată cu succes şi „no" în caz contrar. Dar pentru a transmite aceste mesaje trebuie să ne conectăm la un server de baze de date, pentru a putea scrie într-o tabelă la care avem acces.

Serverul băncii dispune de o bază de date cu trei tabele. Tabela **clienti** conţine numele de cont, seria de card pentru fiecare client al băncii. Tabela **situri** stochează un nume de cont pentru fiecare sit care este client al băncii şi datele necesare conectării serverului de baze de date (server, cont, parolă) al aplicaţiei. O dată conectaţi la acest server mesajele de confirmare vor fi scrise în tabela **transfer** din baza de date a sitului. A treia tabelă, numită **transfer**, conţine informaţiile transferului curent, dar şi informaţiile transferurilor încheiate cu insucces.

Următorul pas este să verificăm dacă seria de card a clientului a fost inserată [illegible] tabela **transfer**. Dacă a fost deja inserată, înseamnă ca transferul a avut loc dar nu [illegible] transmis mesajele de confirmare şi se încearcă o nouă finalizare a transferului. [illegible] seria de card nu există, atunci acesta este un nou transfer şi vom introduce seria [illegible] clientului, suma transferată şi tipul mesajul („yes" sau „no") în tabela

[illegible] este transferul dintr-un cont în altul. După care scriem într-o [illegible] de confirmare („yes"), la fel scriem şi în tabela **transfer** de pe [illegible] a şti dacă am făcut transferul sau nu.

[illegible] al sitului pentru a verifica transferurile încheiate. Adică se [illegible] de pe serverul sitului mesajul de confirmare este „yes". Dacă

este, atunci situl va transmite un mesaj la serverul băncii, cum că a primit mesajul de confirmare.

Se va apela un script al băncii care va verifica mesajul primit de la sit. Dacă mesajul este valid, atunci se va finaliza transferul, iar datele scrise în cele două tabele de pe cele două servere vor fi şterse.

Dacă apar erori în transmiterea de mesaje sau cu conexiunea la serverele de baze de date, atunci datele nu vor fi şterse din tabele. Ele vor fi păstrate până la o nouă încercare de încheiere cu succes a transferului.

Vom prezenta în continuare câteva erori ce pot apărea în timpul unui transfer şi maniera de funcţionare a aplicaţiei în fiecare situaţie.

Plecăm de la presupunerea că avem situaţia în care clientul magazinului virtual are un card bancar cu o valoare de 10.000$. Costul total al produselor selectate este de 5.000$. Clientul începe validarea transferului.

Eroarea 1: Nu se poate conecta la serverul băncii.

Explicaţie: Clientul va fi rugat să încerce mai târziu, iar coşul cu produse va fi păstrat.

Eroarea 2: S-a conectat la serverul băncii, s-a efectuat transferul dintr-un cont în altul, dar s-a întrerupt conexiunea cu serverul sitului.

Explicaţie: Datele transferului sunt salvate în tabela **transfer**, de pe serverul băncii. Clientul va fi informat de eroarea apărută şi va fi rugat să încerce transferul mai târziu. Clientul se poate conecta mai târziu la serverul sitului, după care va încerca să se conecteze la serverul băncii.

Foarte important este faptul că s-ar putea încerca să se modifice coşul de cumpărături, fiind adăugate noi produse în el, în speranţa clientului de a alege mai multe produse şi de a plăti mai puţin. Acest lucru nu se poate întâmpla, pentru că în momentul transferului dintre cele două conturi în tabela **transfer** a fost scrisă suma de transfer. Deci, noua sumă transmisă de client va fi ignorată, până la încheierea cu succes a transferului. Serverul băncii va încerca să transmită din nou mesaj de confirmare la sit.

Eroarea 3: Valoarea transferului a fost de 10.000$ la fel ca şi valoarea cardului bancar. Ce se întâmplă dacă apare o eroare de comunicare, după transferul sumei dintr-un cont în altul? Clientul va fi declarat nesolvabil?

Explicaţie: Se urmează aproape acelaşi mecanism ca la eroarea de mai sus, dar clientul nu va fi declarat nesolvabil deoarece suma a fost transferată în contul sitului.

Eroarea 4: Valoarea transferului este mai mare decât valoarea existentă pe card.

Explicaţie: Clientul va fi declarat ca fiind nesolvabil.

3.7. Căutarea produselor după anumite criterii

O caracteristică esenţială a aplicaţiilor dinamice este găsirea informaţiilor după anumite criterii. Când ne referim la o aplicaţie de tip *e-shop* este necesar să putem căuta informaţia dorită foarte uşor.

Multe aplicaţii de acest tip preferă să implementeze un motor de căutare foarte simplu. Un formular conţinând două câmpuri, un câmp de tip *text* şi [illegible]

submit. Valoarea introdusă în formular este transmisă unei funcţii care caută în baza de date o înregistrare care să conţină valoarea căutată. Este şi acesta un motor intern de căutare, dar facilităţile oferite sunt foarte puţine. Motorul de căutare implementat în cadrul aplicaţiei noastre încearcă să pună la dispoziţie mai multe servicii decât oferă în general aplicaţiile similare.

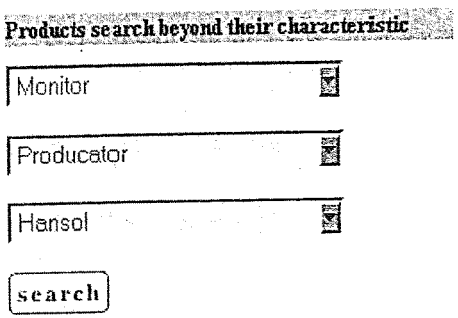
Să presupunem că dorim să căutăm un monitor de culoare roşie. Un motor de căutare fără o ierarhie a categoriilor de produse disponibile poate să ofere multe rezultate care nu-l interesează pe client. La culoarea roşie putem asocia o carcasă a unui sistem, un *mouse* sau – dacă folosim opţiunea de căutare într-o aplicaţie de tip *e-mall* – putem avea surpriza ca în loc de monitoare să găsim produse care nu au nici o legătură cu ceea ce căutăm noi (cămăşi, pixuri, jucării etc.).

Deci implementarea unui motor de căutare – ierarhizat pe categorii de produse, pe caracteristicile acestora, precum şi a valorilor caracteristicilor – este esenţială, dacă dorim ca numărul clienţilor *on-line* să fie într-o ascensiune continuă.

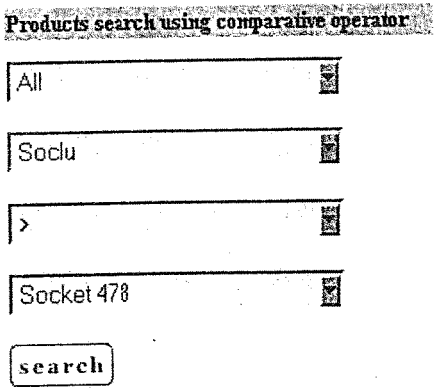
Căutarea după valoarea caracteristicilor este folosită atunci când precizia căutării se doreşte a fi mai mare. De exemplu, căutăm în categoria *Monitoare* toate monitoarele care au ca producător firma **Hansol**.

Cum vom folosi acest tip de căutare? Vom selecta mai întâi categoria de produse unde dorim să căutăm. Această categorie are anumite caracteristici. Una dintre ele este şi **producătorul** produsului. Caracteristica după care vom căuta va fi **producător**. Din lista de valori pentru producător alegem **Hansol**. Nu ne mai rămâne decât să folosim butonul **search**, pentru a obţine rezultatele căutării.

Atunci când folosim al doilea tip de căutare, cel cu operatorii de comparaţie, avem câteva facilităţi în plus faţă de primul tip de căutare. Putem căuta în toate categoriile de produse, putem selecta o caracteristică din cele existente după care un operator de comparaţie. În acest mod căutarea acoperă o arie mai mare de produse care pot îndeplini criteriile căutării.

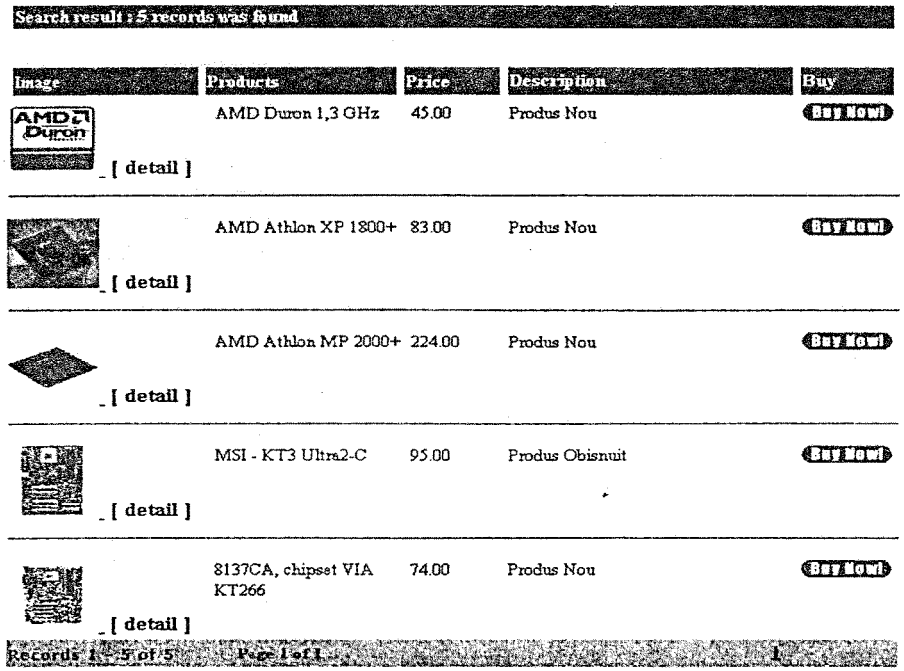


Căutarea produselor folosind operatorii de comparaţie



Căutarea produselor după caracteristici

Figura de mai jos reprezintă rezultatele pentru al doilea tip de căutare, folosind operatorii de comparaţie.



Rezultatele returnate în urma căutării

3.8. Administrarea aplicaţiei

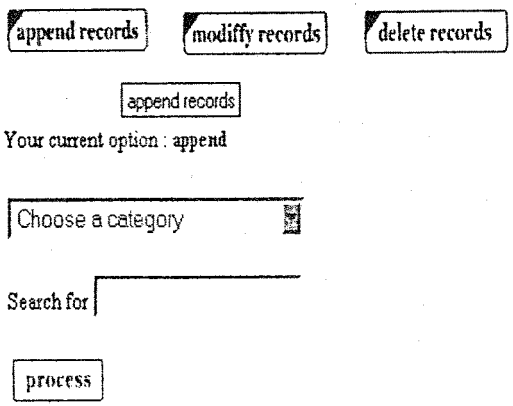
Adăugarea înregistrărilor

Eficienţa şi viteza administrării unei aplicaţii sunt elemente esenţiale pentru succesul magazinelor virtuale. O primă componentă a administrării aplicaţiei este adăugarea unui nou produs într-o anumită categorie sau introducerea unei noi categorii.

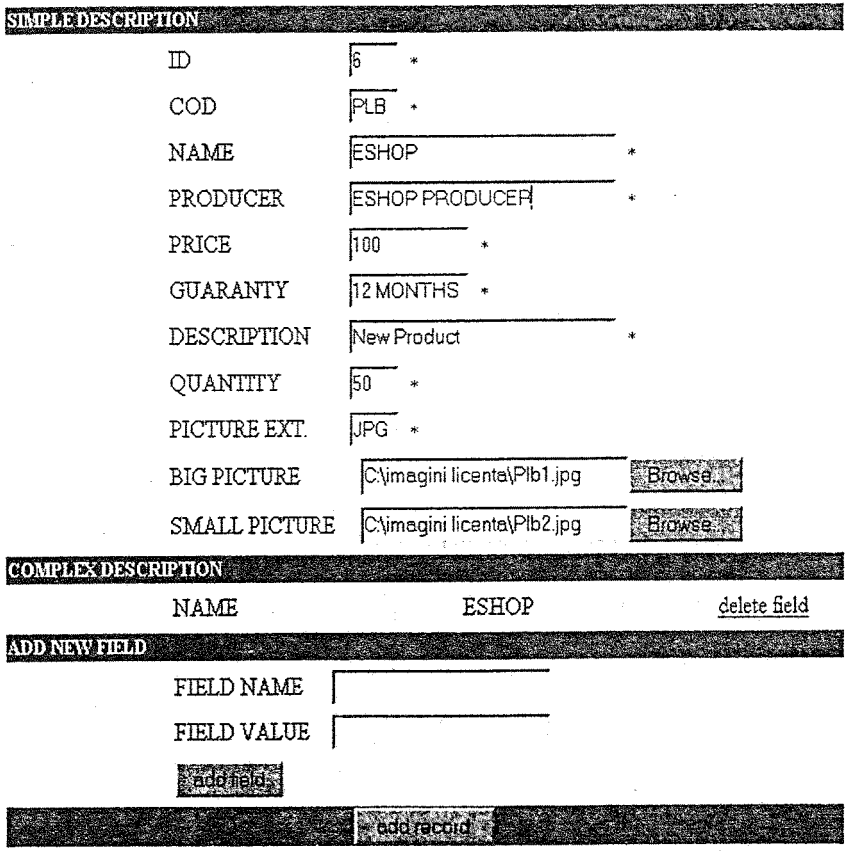
Ce se întâmplă cu modul de afişare al categoriilor, atunci când adăugăm una nouă? Afişarea produselor este dinamică şi din acest motiv se va adăuga, în lista de categorii, noua categorie dorită. Se va genera o nouă coloană care va conţine o legătură spre categoria respectivă.

Figura de mai jos reprezintă serviciile oferite de administrarea aplicaţiei. Prima opţiune este adăugarea unui nou produs sau a unei noi categorii. Se observă „aprinderea" butonului **append records**, în momentul în care dorim să selectăm această opţiune.

Iniţial nu este selectată nici o opţiune.



Utilizarea opţiunii de adăugare de produse



Formularul de introducere a unui produs

Se poate folosi, de asemenea, şi opţiunea de căutare simplă pentru a ajunge la o anumită categorie sau la un anumit produs.

Captura-ecran anterioară reprezintă un formular folosit pentru adăugarea unui nou produs în categoria destinată plăcilor de bază: **Mother Board**. Prima parte a formularului reprezintă o descriere specifică oricărui tip de produs indiferent de categoria din care face parte. A doua parte reprezintă caracteristicile produsului care vor fi afişate atunci când clientul va dori să vadă detaliile tehnice. Această descriere poate fi foarte complexă, modul de completare a ei fiind însă foarte simplu. Pentru fiecare caracteristică putem introduce numele ei şi valoarea. Dacă dorim să eliminăm o astfel de pereche putem folosi legătura **delete field**.

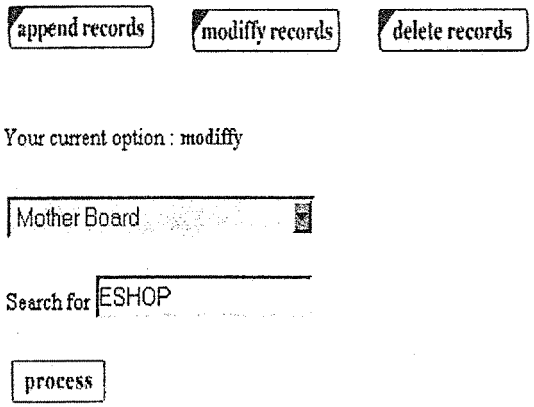
Dacă datele introduse sunt valide şi sunt completate toate câmpurile obligatorii (cele marcate cu steluţă roşie), atunci vom avea un nou produs în lista de produse.

Modificarea înregistrărilor

Al doilea serviciu oferit de acest modul de administrare este modificarea informaţiilor unei anumite înregistrări.

Cum putem realiza acest lucru? Mai întâi trebuie selectată opţiunea de modificare şi anume butonul **modify records**. După selectarea acestei opţiuni, aplicaţia ne informează despre opţiunea curentă. Următorul pas este selectarea unei categorii sau căutarea unui produs dintr-o anumită categorie. Utilizăm butonul **process** pentru a obţine produsele sau produsul care urmează a fi modificat.

Prezentăm mai jos un exemplu de căutare a unui produs şi de modificare a acestuia. Presupunem că dorim să modificăm produsul pe care deja l-am inserat când am folosit opţiunea de adăugare. Selectăm categoria **Mother Board**, după care vom introduce cuvântul-cheie ESHOP care reprezintă numele produsului.



Căutarea unui produs pentru a fi modificat

Rezultatul căutării este prezentat în imaginea de mai jos:

Search result : 1 records was found

Image	Products	Price	Description	Option
	ESHOP	100.00	New Product	modify

Records 1 - 1 of 1 Page 1 of 1

Rezultatul căutării pentru opţiunea de modificare

După ce am găsit produsul nu ne mai rămâne decât să deschidem formularul de modificare pentru produsul căutat, printr-un simplu *click* pe butonul **modify** sau un *click* pe imaginea produsului.

Spre deosebire de alte module de administrare, cel prezentat aici permite o modificare totală a descrierii tehnice a produsului, în sensul că pot fi modificate atât valorile caracteristicilor, cât şi numele lor.

Formularul de mai jos reprezintă opţiunea de modificare a modulului de administrare.

SIMPLE DESCRIPTION

ID	89 *
COD	plb *
NAME	ESHOP *
PRODUCER	ESHOP PRODUCER *
PRICE	100.00 *
GUARANTY	12 MONTHS *
DESCRIPTION	New Product *
QUANTITY	50 *
PICTURE EXT.	jpg *
BIG PICTURE	Browse
SMALL PICTURE	Browse

COMPLEX DESCRIPTION

NAME	ESHOP
PRICE	100

Formularul de modificare a informaţiilor despre un produs

Ştergerea înregistrărilor

Ultima opţiune a modului de administrare este ştergerea unui produs dintr-o anumită categorie. La fel ca la opţiunea de modificare, mai întâi se schimbă opţiunea curentă, folosind butonul **delete records**, după care selectăm o categorie sau folosim opţiunea de căutare pentru a localiza un produs, ultimul pas fiind selectarea butonului **delete**.

Concluzii

Pe parcursul acestui subcapitol am discutat despre divizarea aplicaţiei în module şi modul de interacţiune dintre ele. Am explicat detaliat opţiunile fiecărui modul, ilustrând cu secvenţe de cod şi capturi-ecran.

Problemele apărute în timpul comunicării dintre module au fost expuse pe larg în secţiunea 3.6, oferindu-se şi soluţiile implementate în această aplicaţie. Ne-am concentrat atenţia spre serviciul de plată prin transfer bancar, dar şi asupra modulelor de căutare, de autentificare şi de administrare a aplicaţiei.

4. Manual de utilizare

În acest capitol ne vom ocupa în mod special de modul de utilizare a aplicaţiei, pornind de la cele mai simple lucruri şi până la situaţii mai complicate cum ar fi transferul bancar şi problemele ce pot apărea.

Vom structura acest capitol în funcţie de ordinea acţiunilor ce vor apărea în cursul navigării. Putem privi aceste acţiuni ca un arbore, care are ca rădăcină pagina de start a sitului. Acţiunile care se pot realiza plecând din acest pas sunt numeroase, apărând în arbore ca noduri-frunză.

Din pagina de start avem următoarele acţiuni:

- autentificarea;
- crearea unui utilizator nou;
- contactarea administratorului;
- selectarea unei categorii de produse;
- selectarea unei opţiuni din meniul din dreapta paginii:
  - harta sitului (*Site Map*);
  - întrebări frecvente (*FAQ*);
  - copierea unui catalog de produse;
  - sponsorul sitului;
  - preferinţele utilizatorului;
  - suport *on-line* pentru ajutor (*Help*);
- copierea kitului de instalare pentru navigatorul *Mozilla* pe baza căruia s-a dezvoltat aplicaţia.

Această clasificare reprezintă doar primul nivel al arborelui. Pentru a nu crea confuzii şi a fi cât mai expliciţi în explicarea acestor opţiuni vom prezenta fiecare acţiune de pe primul nivel în detaliu.

Autentificarea reprezintă completarea de către utilizator a câmpurilor **user** şi **passwd**, după care se va acţiona butonul **login**. Dacă datele sunt valide vizitatorul va fi redirectat la o nouă pagină care reprezintă contul acestuia.

După cum am văzut şi în capitolul precedent, pot apărea erori dacă datele introduse sunt invalide. Datele invalide reprezintă date inexistente, parolă sau nume de utilizator incorecte.

Acţiunile care se pot realiza de la acest pas sunt următoarele:

• *ieşirea din cont*

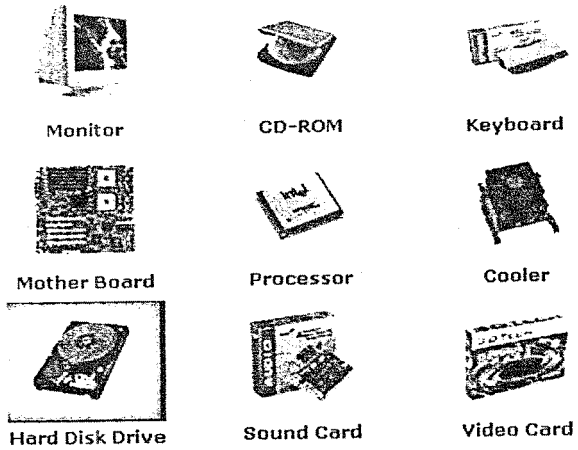
Ce reprezintă această opţiune şi de ce trebuie să o folosim? Reprezintă acţiunea prin care utilizatorul doreşte să părăsească aplicaţia sau să nu mai efectueze acţiuni asupra contului său dorind să fie utilizator anonim.

• *contactarea administratorului*

Această acţiune se foloseşte atunci când apare o problemă care nu este explicată în suportul de ajutor *on-line* sau în secţiunea de întrebări frecvente. Pentru a beneficia de acest serviciu, utilizatorul trebuie să aibă configurat un serviciu de *e-mail*, *Outlook Express* de exemplu. Dacă nu este folosit un astfel de serviciu, utilizatorul poate observa adresa de contact în bara de stare atunci când *mouse*-ul este deasupra butonului contact şi o poate folosi pentru a contacta administratorul aplicaţiei.

• *selectarea unei categorii de produse*

Selectarea unei categorii de produse se realizează activând un idiom corespunzător categoriei de produse. Atunci când focusul *mouse*-ului este pe un astfel de idiom, idiomul respectiv „se aprinde" sau textul care indică numele categoriei de produse îşi schimbă culoarea.



Selectarea categoriei Hard Disk Drive

• *coşul de cumpărături*

O altă opţiune este coşul de cumpărături care se activează atunci când folosim butonul *Your Cart*. Acţiunile care se pot efectua asupra coşului de cumpărături le-am amintit şi în subcapitolul precedent: modificarea cantităţii pentru fiecare produs în parte, eliminarea unui produs, calculul automat al preţului total, adăugarea de noi produse, transferul bancar etc.

• *ştergerea coşului de cumpăturí*

Această opţiune şterge toate produsele din coş.

• *căutarea*

Am vorbit în detaliu despre această facilitate în capitolul precedent în secţiunea „Căutarea produselor după anumite criterii".

• *selectarea unei opţiuni din meniul din dreapta paginii*

Aceste opţiuni au rolul de a informa utilizatorul şi de al ajuta atunci când nu găseşte răspunsul la o întrebare. Figura alăturată reprezintă opţiunile meniului despre care discutăm.

site services

site map
faq
catalog download
sponsor
preferences
getting help

Meniu de legături rapide

• *copierea kitului de instalare a navigatorului Mozilla*

Această aplicaţie a fost concepută iniţial pentru navigatorul **Mozilla 1.2.1**, fiind *browser*-ul recomandat pentru vizualizarea aplicaţiei. Situl suportă şi celelalte tipuri de navigatoare cu precizarea că pentru fiecare în parte trebuie precizat un anumit fişier CSS de formatare a datelor. Astfel, am conceput un fişier de formatare a datelor şi pentru *Internet Explorer 6.0*, astfel încât utilizatorul poate folosi oricare dintre aceste două navigatoare Web.

Crearea unui utilizator nou reprezintă opţiunea prin care un client care nu este înscris în baza de date a aplicaţiei poate completa un formular pentru a deveni un membru activ al aplicaţiei. Datele obligatorii pentru înscrierea unui nou utilizator sunt: **account, password, last name, first name, e-mail, bank**. Celelalte câmpuri din formular sunt opţionale.

Se pot ivi probleme dacă datele introduse sunt invalide. Atunci când alegem un nume de utilizator deja existent, când nu completăm toate câmpurile obligatorii sau când formatul adresei de *e-mail* este incorect vom fi avertizaţi prin mesaje de eroare. Imaginea de mai jos exprimă cele discutate mai sus:

Account	eshop	*
Password	**********	*(6-30 characters)
Last name	ciprian	*
First name	ginghina	*
Ocupation	student	
Email	e@shop.ro	*
Sex	M	
Status		
City		
Residence		
Phone		
Bank	BCP	*
	submit	

Formularul de înscriere

Atunci când numele de cont există deja în baza de date vom primi un mesaj de avertizare, iar utilizatorul va trebui să aleagă alt nume.

Contactarea administratorului aplicaţiei atunci când apare o eroare pe care utilizatorul nu poate să o rezolve a fost explicată mai sus.

Selectarea unei categorii de produse este o facilitate despre care am discutat, de asemenea, anterior.

Selectarea unei opţiuni din meniul din partea dreaptă a aplicaţiei, reprezintă legăturile pe care utilizatorul le poate folosi pentru a se putea informa sau pentru a-şi selecta preferinţele de afişare a înregistrărilor.

- **harta sitului** reprezintă cele mai importante legături din cadrul aplicaţiei;
- **întrebări frecvente** este o secţiune destinată problemelor care pot apărea în timpul navigării, informării utilizatorilor despre condiţiile de livrare a produselor etc. Această secţiune se poate actualiza de către administratorul sitului;
- **copierea catalogului de produse** reprezintă salvarea pe calculatorul local al utilizatorului a unui fişier cu informaţii despre produsele existente;
- **sponsorul aplicaţiei** poate fi o aplicaţie similară, o persoană fizică, o firmă etc.
- **preferinţele utilizatorului** este o opţiune prin care clientul poate alege câte produse să fie afişate pe pagină, numărul de legături pe pagină, un anumit stil de formatare a datelor ce urmează a fi afişate;
- **suportul *on-line* pentru ajutor** reprezintă o versiune în format HTML a actualului manual de utilizare.

Multe dintre acţiunile amintite se repetă în unele noduri ale arborelui de acţiuni corespunzător aplicaţiei noastre. Vom prezenta în continuare altele: navigarea în

cadrul catalogului de produse, transferul bancar şi modificarea coşului de cumpărături.

Navigarea în cadrul catalogului de produse este extrem de intuitivă şi poate fi modificată după dorinţele utilizatorului. Atunci când deschidem aplicaţia sau ne autentificăm, categoriile de produse disponibile sunt afişate în partea centrală a paginii. Putem selecta oricare din ele pentru a vedea ce produse conţine. La fiecare pas al navigării putem schimba numărul de produse afişate pe pagină, stilul de vizualizare, precum şi numărul de legături destinate răsfoirii listei.

Pe parcursul navigării putem adăuga în coş produsele dorite şi avem posibilitatea revenirii la lista de categorii folosind butonul **Products**. Vizualizarea coşului de cumpărături se realizează cu butonul **Your Cart**, având de asemenea posibilitatea ştergerii conţinutului său.

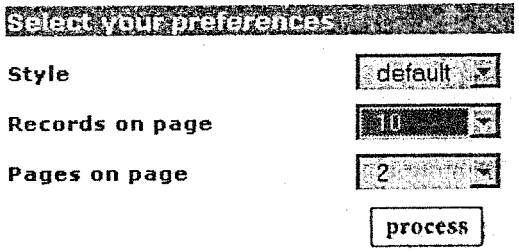
Prezentăm două capturi-ecran care exprimă acţiunile oferite clientului de către aplicaţie. Prima imagine se referă la navigarea în cadrul catalogului de produse, precum şi la serviciile oferite de această navigare: detalii despre produs, posibilitatea adăugării în coş şi vizualizarea restului de produse.

Image	Products	Price	Description	Buy
_[detail]	HANSOL H530	349.00	Produs Nou	Buy Now!
_[detail]	SDM S51	479.00	Produs Obisnuit	Buy Now!
_[detail]	CTX PV 720	785.00	Produs Nou	Buy Now!
_[detail]	SDM S51B	499.00	Produs Nou	Buy Now!
_[detail]	IESOLAR GP770	189.00	Produs Obisnuit	Buy Now!

Records 16 - 20 of 22 Page 4 of 5 3 4

Navigarea în catalogul de produse

Cea de-a doua imagine ilustează posibilitatea clientului de a-şi modifica propriile preferinţe:

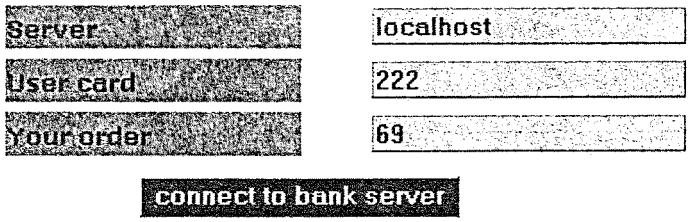


Preferinţele utilizatorului

Transferul bancar se realizează atunci când clientul doreşte să plătească pentru produsele solicitate. Atunci când vizualizăm conţinutul coşului avem posibilitatea să iniţiem procesul de transfer bancar prin intermediul unui card bancar.

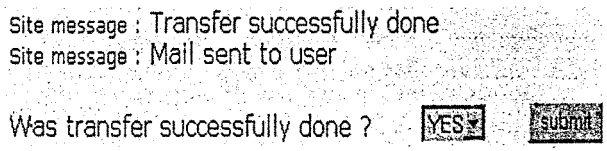
Pentru a începe acest proces folosim butonul **send the order**. Se va afişa un formular în care sunt completate două din cele trei câmpuri existente. Primul câmp se referă la adresa serverului băncii, al doilea – care va fi completat de către client – reprezintă seria cardului bancar, iar ultimul câmp conţine valoarea sumei transferate.

Vom folosi butonul **connect to bank server** pentru a realiza acest transfer. Dacă datele introduse sunt valide, adică dacă seria de card este corectă şi clientul este solvabil, atunci se va efectua transferul. O eroare la acest pas conduce la apariţia unui mesaj de avertizare şi la încheierea acestei acţiuni.



Conectarea la serverul băncii

Dacă nu apare nici o eroare şi transferul a avut loc cu succes, utilizatorul va răspunde la o întrebare al cărei răspuns este completat automat în funcţie de rezultatul transferului până la acest pas. Suntem întrebaţi de aplicaţie dacă transferul a avut loc cu succes. Dacă răspunsul este **Yes** putem folosi butonul submit pentru încheierea transferului. În caz contrar răspunsul va fi **No** şi clientul va trebui să încerce mai târziu finalizarea transferului.



Încheierea transferului

Modificarea coşului de cumpărături se referă la acţiuni despre care am discutat deja: adăugarea unor noi produse în coş folosind legătura **Keep Shopping**, ştergerea unei înregistrări din coş folosind butonul **Remove**, modificarea cantităţii produselor selectate şi calcularea automată a costului total.

5. La final...

Capitolul de faţă a prezentat unele aspecte interesante ale procesului de dezvoltare a unei aplicaţii Web de tip *e-shop*, stabilă, ergonomică, sigură şi cât mai aproape de nevoile utilizatorului. Am vorbit despre instrumentele software folosite şi am argumentat avantajele soluţiei alese comparativ cu alte soluţii existente pe piaţă.

În continuare s-a trecut la analiza aplicaţiei, clasificarea ei în module şi maniera de interacţiune a acestora. Am analizat problemele care pot apărea în cadrul acestor module şi am propus, sperăm, soluţii cât mai eficiente. Am încheiat această secţiune cu prezentarea succintă a celor mai importante cerinţe ale aplicaţiei.

În următorul subcapitol am discutat despre detalii de implementare, explicându-se maniera de concepţie a unor module din aplicaţie şi prezentându-se secvenţe de cod-sursă.

În final, s-a pus la dispoziţie o secţiune despre modul de utilizare a aplicaţiei, încercând să soluţionăm eventuale problemele ce pot surveni pe parcursul exploatării sitului.

Cititorii interesaţi pot experimenta implementarea unui magazin virtual pornind de la cele prezentate în cadrul acestei lucrări, putând – de asemenea – să îmbunătăţească aplicaţia prin adăugarea unor module noi. De exemplu, prezentarea produselor s-ar putea realiza prin intermediul unor tehnici tridimensionale (a se vedea şi cele discutate în capitolul 5 al cărţii de faţă).

Resurse bibliografice

O serie de referinţe bibliografice utile pe care le poate consulta şi aprofunda cititorul interesat sunt următoarele:

1. D. Acostăchioaie, *Securitatea sistemelor Linux*, Editura Polirom, Iaşi, 2003
2. S. C. Buraga, *Tehnologii Web*, Editura MatrixRom, Bucureşti, 2001
3. S. C. Buraga, *Proiectarea siturilor Web*, Editura Polirom, Iaşi, 2002
4. S. C. Buraga, V. Tarhon-Onu, Ş. Tănasă, *Programare în bash şi Perl*, Editura Polirom, Iaşi, 2002
5. T. Bidwell *et al.*, *Hack Proofing Your E-commerce Site*, Syngress Publishing, 2001
6. M. Cioată, „Plăţi electronice", *NET Report*, feb. 2001
7. M. Fotache, „Proiectarea bazelor de date", *NET Report*, mart. 2003
8. T. Gerken, T. Rastchiller, *Web Aplication Development with PHP 4.0*, New Riders Publishing, Indianapolis, 2000
9. F. Pascal, „Ce-i greşit în peisajul bazelor de date", *NET Report*, iun. 2002
10. M. Sabău, „Tunelarea", *NET Report*, dec. 2002

11. R. Stoian, „Siguranţa serviciilor Web", *PC Report*, nov. 2000
12. M. Târnovan, Ş. Târnovan, „Un exemplu practic privind implementarea unui magazin virtual", *NET Report*, dec. 2002
13. * * *, *Apache*: http://www.apache.com
14. * * *, *Ergonomia în designul paginilor Web*, Edo2000, apr. 1999
15. * * *, *MySQL Reference Manual*: http://www.mysql.com
16. * * *, *PHP Classes*: http://www.classes.net

CAPITOLUL 4

E-Pizza – de la browser la WAP

Cristian Nechita, angel@infoiasi.ro
Marius Ropotă, ropo@infoiasi.ro
Iavi Rotberg, pit@infoiasi.ro

Aplicaţia *E-Pizza* este o soluţie completă pentru managementul informaţiilor referitoare la activitatea unei pizzerii, incluzând comandarea *on-line* de produse, rezervarea de mese în restaurant şi administrarea facilă prin intermediul unei interfeţe Web. De asemenea, se descrie şi o soluţie de accesare – via dispozitive WAP – a sitului, utilizându-se limbajul WML.

1. Descriere generală

Capitolul de faţă prezintă modul în care a fost gândită şi implementată aplicaţia *E-Pizza*. Partea de aplicaţie destinată navigatoarelor obişnuite a fost implementată în mare parte în PHP, folosindu-se JavaScript pentru un modul destinat rezervărilor. Partea care este destinată dispozitivelor *WAP* (*Wireless Application Protocol*) utilizează *WML* (*Wireless Markup Language*) pentru interfaţa cu utilizator. Pentru a asigura o compatibilitate mai bună cu aplicaţiile de tip *WAE* (*WAP Application Enviroment*) sau cu aşa-numitele *navigatoare WAP* s-a evitat folosirea scripturilor de tip WMLScript, folosindu-se limbajul PHP pentru un control elaborat al prezentării pe spaţiul client.

Pentru început vor fi enunţate o parte dintre obiectivele pe care şi le propune să le îndeplinească aplicaţia, urmând ca apoi să se stabilească principalele cerinţe şi să se analizeze modul de dezvoltare al modulelor componente. Capitolul continuă cu descrierea arhitecturii software a aplicaţiei, oferind cele mai reprezentative funcţii implementate. Spre final prezentăm un scurt ghid de utilizare.

1.1. Obiective

Aplicaţia *E-Pizza* îşi propune să realizeze managementul unui restaurant de tip pizzerie, facilitând comandarea de pizza, respectiv plata electronică a comenzii, precum şi rezervarea meselor din cadrul restaurantului.

Pentru realizarea managementului restaurantului şi a celorlalte operaţii (comandă, rezervare, plată) se va proiecta un sit Web. Prin intermediul acestui sit Web sau prin intermediul unui dispozitiv mobil (utilizând tehnologia WAP) vor fi disponibile toate facilităţile pe care le-au vizat obiectivele aplicaţiei.

În ceea ce priveşte comanda electronică se vor oferi următoarele facilităţi:

- clienţii vor putea opta între două moduri de comandă: *comandă rapidă* sau *comandă autentificată*;
- posibilitatea de a comanda un sortiment de pizza nou, adică o pizza creată în funcţie de preferinţele clientului;
- clientul va putea vizualiza pe perioada interacţiunii cu situl informaţii privitoare la comanda sa (factura detaliată, cantitatea şi tipul fiecărei pizza comandate);
- înştiinţarea clientului prin intermediul unui SMS asupra acceptării comenzii.

Pentru rezervarea electronică a unei mese din restaurant se vor avea în vedere următoarele:

- posibilitatea clienţilor de a rezerva/anula o masă în orice zi din săptămâna în curs;
- vizualizarea situaţiei ocupării meselor din săptămâna în curs;
- posibilitatea oferită managerului restaurantului de a urmări în timp real aranjamentul meselor din cadrul restaurantului, precum şi numărul acestora.

Cu privire la plata electronică se va avea în vedere în primul rând realizarea unui sistem cât mai fiabil pentru efectuarea tranzacţiilor dintre client şi restaurant prin intermediul unei bănci.

1.2. Cerinţe

În proiectarea şi dezvoltarea aplicaţiei vor trebui rezolvate următoarele specificaţii de cerinţe:

- posibilitatea utilizatorului de a-şi crea un cont pe server cu ajutorul căruia se va autentifica ulterior;
- posibilitatea utilizatorului de a modifica informaţiile referitoare la contul său;
- prezentarea meniului zilei cu oferirea facilităţii de ordonare după diverse criterii, precum numele pizzei, greutate, mărime sau preţ;
- vizualizarea comenzii în mod dinamic, astfel încât clientul să aibă posibilitatea de a adăuga sau elimina un anumit produs în funcţie de preferinţe;
- prezentarea în mod grafic, dar şi textual, a unui anumit produs selectat de utilizator la un moment dat;
- posibilitatea de a rezerva o anumită masă la o anumită dată;
- administrarea listei de bunuri comercializate, ca de exemplu posibilitatea de a adăuga sau şterge ingrediente şi produse;
- posibilitatea de a vizualiza şi şterge anumite comenzi (autentificate sau nu);

- posibilitatea de a oferi în mod dinamic prezentări de informaţii manageriale (vânzări, profit etc.) care să reflecte activitatea restaurantului;
- posibilitatea de a modifica numărul de mese, precum şi configuraţia acestora în orice zi din săptămâna în curs;
- posibilitatea de a vizualiza în orice moment situaţia meselor (care sunt rezervate şi care nu, precum şi diverse detalii despre o rezervare);
- posibilitatea de a genera o configuraţie aleatorie pentru mesele din restaurant;
- realizarea unei interfeţe care să ofere consistenţă precum şi o operare simplă din punctul de vedere al utilizatorului.

1.3. Analiza aplicaţiei

În vederea soluţionării problemei propuse am împărţit aplicaţia în următoarele module:

- Comanda *on-line*
- Administrarea sitului
- Grafice
- Rezervarea *on-line*
- Plata electronică
- Comanda via WAP

În cele ce urmează vom descrie funcţionalităţile oferite de fiecare dintre aceste module. Vom prezenta şi o serie de aspecte legate de implementarea efectivă a modulelor respective.

Modulul pentru efectuarea de comenzi on-line

Acest modul va permite atât utilizatorilor înregistraţi, cât şi celor neînregistraţi să comande pizza prin intermediul sitului. De asemenea permite unui administrator să vizualizeze şi să valideze aceste comenzi. Având în vedere cerinţele aplicaţiei referitoare la modulul de faţă, acesta va fi împărţit în două submodule:

- Comenzi autentificate
- Comenzi rapide

Pentru implementarea submodulului *Comenzi autentificate* va trebui, în primul rând, rezolvată problema autentificării clienţilor, aceasta constituind şi diferenţa majoră dintre aceste două submodule. Astfel, fiecare client se va autentifica printr-un nume (pseudonim) unic şi o parolă care va fi păstrată criptată în baza de date (a se vedea şi capitolul 2). Acest mod de structurare a bazei de date va oferi un grad de securitate mai înalt şi un acces mai rapid la date datorită faptului că în momentul efectuării unei tranzacţii se va lucra numai cu numele clientului, restul informaţiilor necesare finalizării operaţiei vor fi obţinute prin intermediul acestui nume.

S-a luat în considerare următoarea structură a bazei de date:

Comenzi Rapide

ID	int(10)
NUME	char(20)
PRENUME	char(30)
ADRESA	char(50)
TEL	int(10)
DATA_C	datetime
ID_PIZZA	int(3)
CANT	int(2)
LIVRATA	int(1)

Comenzi Autentificate

ID	int(10)
USER	char(12)
DATA_C	datetime
ID_PIZZA	int(3)
CANT	int(2)
LIVRATA	int(1)
PLATITA	int(1)

Structurile bazelor de date utilizate pentru stocarea comenzilor

Modulul de administrare a sitului Web

Acest modul va permite administratorului să modifice atât forma sitului, cât şi conţinutul acestuia, precum şi să vizualizeze anumite informaţii. Utilizatorii înregistraţi vor putea să-şi exprime părerea în legătura cu serviciile *E-Pizza* prin intermediul unor chestionare disponibile *on-line*.

Datorită complexităţii acestui modul s-a recurs la divizarea lui în cinci submodule: *utilizatori*, *pizza*, *ingrediente*, *noutăţi*, *chestionare*. Pe baza acestor submodule au fost proiectate şi structurile tabelelor *utilizatori* şi *pizza* după cum reiese şi din imaginea de mai jos.

Utilizatori

USER	char(12)
PASSWD	char(12)
NUME	char(20)
PRENUME	char(30)
ADRESA	char(50)
TEL	bigint(10)
MOBIL	bigint(10)
EMAIL	char(30)
CARD_NO	bigint(16)
TIP	char(15)
EXP_DATE	date

Pizza

ID_PIZZA	int(3)
NUME_P	char(25)
MARIME	int(2)
GREUTATE	int(3)
PRET	int(6)
INGREDIENTE	char(50)
VEGET	int(1)
POZA	char(30)
RETETA	char(30)

Structurile bazelor de date stocând date despre utilizatori şi sortimentele de pizza

Administratorul va putea modifica:

- noutăţile prezente în cadrul sitului;
- meniul restaurantului;
- ingredientele folosite la prepararea pizzei;
- utilizatorii înregistraţi.

Administratorul va putea vizualiza o serie de grafice (generate dinamic) ce vor reflecta:

- desfăşurarea activităţilor din cadrul restaurantului;
- părerea utilizatorilor despre serviciul *E-Pizza*.

Utilizatorii înregistraţi îşi vor putea exprima părerea completând cel mult o dată pe zi un chestionar pus la dispoziţie de sit. Pentru a oferi o imagine cât mai fidelă asupra serviciilor oferite de *E-Pizza* va trebui implementat un mecanism care să garanteze că un anumit client nu va vota de mai multe ori pe zi.

Pentru a evita supraîncărcarea bazei de date, aplicaţia va trebui să ofere administratorului o modalitate de a şterge conturile care nu au fost folosite într-un anumit interval.

Modulul de generare a graficelor

Acest modul oferă pe de o parte administratorului de restaurant informaţii cu privire la preferinţele clienţilor în privinţa meniului zilnic (prin comenzile efectuate de aceştia), precum şi *feedback*-ul dat de clienţi prin completarea chestionarelor *on-line*.

Graficele trebuie să reflecte următoarele chestiuni legate de funcţionarea serviciului *E-Pizza*:

- comparaţii între numărul de comenzi pe care le efectuează utilizatorii înregistraţi, faţă de cei neînregistraţi;
- cele mai comandate pizza prin comenzi autentificate;
- cele mai comandate pizza prin comenzi rapide;
- comparaţii ale vânzărilor pe luni;
- comparaţii ale comenzilor pe zilele săptămânii (cele mai solicitate zile);
- cele mai folosite ingrediente (pentru comezile autentificate şi cele rapide).

Modulul de rezervări on-line

După cum am precizat şi mai sus, acest modul va permite unui utilizator să-şi rezerve una sau mai multe mese în restaurant pentru o anumită dată. De asemenea, modulul poate fi folosit şi de către administratorul sitului pentru a modifica anumite date legate de restaurant, el putând, de exemplu, să schimbe aranjamentul meselor de pe pagină pentru a reflecta maniera în care mesele sunt dispuse în restaurant. Mai mult, administratorul are posibilitatea de a adăuga sau de a şterge mese din restaurant (acest lucru fiind o reacţie la faptul că în restaurant numărul de mese a crescut, respectiv a scăzut, datorită anumitor factori externi). Având în vedere cele discutate mai sus rezultă că acest modul va fi împărţit în două submodule:

• *modulul administrator*

Rolul acestui modul este acela de a permite unui utilizator să poată vizualiza amplasamentul real al meselor din restaurant. Acest lucru este necesar deoarece unii dintre utilizatori vor avea preferinţe asupra amplasamentului mesei pe care o vor rezerva. Deci este imperios necesar ca amplasamentul meselor afişat pe sit să corespundă cu cel din realitate.

Datorită unor anumiţi factori, aranjamentul meselor din restaurant se poate modifica, revenind astfel administratorului sarcina de a actualiza pagina cadrul sitului corespunzătoare rezervării *on-line*. O altă situaţie care ar putea apărea este modificarea numărului de mese de care dispune restaurantul la un moment dat. Nu este permis ca pe pagina Web să fie prezent un număr mai mare de mese (acest lucru conduce la efectuarea de rezervări care nu vor putea fi onorate şi poate chiar la pierderea de potenţiali clienţi). Pe de altă parte, dacă pe pagina Web sunt prezente mai puţine mese decât deţine în mod curent restaurantul, apare posibilitatea ca acele mese care nu sunt prezente pe sit să rămână neocupate şi astfel pot interveni pierderi financiare în cadrul firmei.

• *modulul client*

Rolul acestui modul este de a permite unui utilizator să rezerve una sau mai multe mese la o anumită dată. Modulul trebuie întâi de toate să ofere utilizatorului aranjamentul real al meselor din restaurant, plus mesele care sunt rezervate şi care nu. Administratorul va asigura consistenţa configuraţiei, adică aranjamentul meselor este cel real şi tot el va fi responsabil pentru anularea rezervărilor persoanelor care nu şi-au „revendicat" rezervarea în timp util. O altă funcţionalitate pe care ar trebui să o ofere acest modul este rezervarea în sine. Modulul trebuie să afişeze datele corecte pentru fiecare masă, prin acestea înţelegându-se întâi de toate figurarea unui marcaj care să evidenţieze mesele rezervate. De asemenea, modulul va trebui să afişeze pentru o anumită masă care a fost în prealabil rezervată perioada de timp pe care a fost rezervată. Acest lucru este util atunci când restaurantul nu mai are mese disponibile, pentru ca utilizatorul să ştie care masă se va elibera şi când anume.

Modulul de plată electronică

Având în vedere faptul că tranzacţiile vor fi efectuate electronic, trebuie implementat un serviciu care să ducă la bun sfârşit tranzacţiile începute, oferind garantarea securităţii datelor „delicate" (de exemplu, seria cărţii de credit) furnizate de către clienţi.

Pentru a asigura securitatea clienţilor vom cripta informaţiile cu privire la cartea de credit, folosind un algoritm care va genera o cheie privată cu o lungime mai mare de 128 biţi (iar dacă informaţiile vor fi păstrate pe o perioadă mai mare de cinci ani este de preferat ca lungimea cheii să depăşească 256 biţi). Cheia privată împreună cu cheia publică vor fi folosite de acest algoritm pentru a cripta informaţiile dorite. A se vedea şi *Securitatea sistemelor Linux* de Dragoş Acostăchioaie.

Modulul de comandare prin WAP

Acest modul va permite utilizatorilor înregistraţi să comande pizza *on-line* (prin suport WAP) efectuând plata acestor comenzi prin cartea de credit, iar celor neînregistraţi să comande pizza realizând plata comenzii la livrare.

Având în vedere obiectivele prezentate mai sus modulul va fi divizat în două submodule pentru procesarea comenzilor autorizate şi a celor rapide.

- **Procesarea comenzilor autorizate**
  Utilizatorul înregistrat va selecta din meniul afişat pe ecranul dispozitivului WAP câte un anumit tip de pizza, plus cantitatea pe care o doreşte, după care va efectua comanda dorită.
  Administratorul se va ocupa de administrarea bazelor de date (ştergerea unei comenzi, onorarea ei şi completarea în tabele a valorilor necesare, administrarea meniului, precum şi a meselor din restaurant), toate acestea prin intermediul unui modul a cărui implementare foloseşte tehnologiile Web tradiţionale (HTTP, HTML şi CSS, cu concursul scripturilor PHP). Precizăm faptul că modulul WAP este creat numai pentru uzul clienţilor.
- **Procesarea comenzilor rapide**
  Utilizatorul neînregistrat îşi va alege din meniul pizzeriei tipul pizzei, cantitatea şi, de asemenea, va completa datele personale: nume, prenume, adresa şi telefonul la care poate fi contactat şi la care se va face verificarea/validarea comenzii.

De remarcat faptul că utilizatorul înregistrat şi autentificat pe sit va putea să-şi vizualizeze informaţiile introduse la înregistrare, având de asemenea şi posibilitatea de a le modifica (de exemplu, schimbarea numărului cărţii de credit sau modificarea adresei).

1.4. Arhitectura software a aplicaţiei

Pentru dezvoltarea acestei aplicaţii s-au folosit următoarele instrumente şi tehnologii software: PHP, JavaScript şi XML.

În continuare vom analiza arhitectura software a fiecărui modul discutat anterior.

Modulul pentru efectuarea de comenzi on-line

Având în vedere funcţionalităţile prezentate la punctul anterior, am împărţit acest modul în două secţiuni: partea administratorului – care va implementa funcţionalităţile cerute de administrator – şi partea utilizator înregistrat – care va implementa funcţionalităţile cerute de utilizatorii înregistraţi.

Având în vedere atributele unei comenzi autentificate, am agregat toate comenzile unui utilizator de la o anumită dată într-o comandă care va putea fi referită printr-o cheie primară compusă, formată din identificatorul utilizatorului (*user*) şi data la care a fost efectuată comanda. Am considerat că această agregare va da mult mai multă libertate administratorului, mărind în acelaşi timp şi eficienţa acestuia.

Această operaţie s-a realizat folosind următoarea funcţie:

```
function getOrders() {
global $link; // variabila globala

$query = "SELECT `USER`, `DATA_C`  FROM `comenzi_a` GROUP
BY `DATA_C` ORDER BY `DATA_C` ASC";

$result = mysql_query($query, $link) or
              die("Error:" . mysql_error());

return $result;
}
```

Pentru a selecta apoi toate comenzile autentificate efectuate de un anumit utilizator înregistrat s-a utilizat funcţia:

```
function getOrder($user, $data) {
global $link;

$query = "SELECT * FROM `comenzi_a` WHERE user='$user' AND
data_c='$data' ";
$result = mysql_query($query, $link) or
              die("Error:" . mysql_error());

return $result;
}
```

Pentru a permite utlizatorului să vizualizeze în orice moment comanda sa, am realizat o clasă denumită *cos*. Această clasă implementează un coş de cumpărături cu ajutorul mecanismului de sesiuni oferit de PHP (a se revedea şi cele discutate în cadrul capitolului 3).

Pentru a înregistra o sesiune am folosit funcţia PHP *bool session_register (mixed name [, mixed ...])*.

Prezentăm în continuare codul clasei *cos* (o soluţie alternativă a fost expusă în capitolul 3):

```
class cos {

// date membre
var $pizza; // tablou asociativ id_pizza=>cantitate
var $value; // valoarea cosului de cumparaturi

// metode implementate
function count_pizza() {
// metoda pentru a calcula nr. total de produse din cos
```



```
     $count=0;

     if ( $this->pizza )
              foreach($this->pizza as $val){
                     $count += $val;
              }
      return $count;
}

function add_pizza($pizza_id, $quant) {
// metoda pentru a adauga o anumita cantitate de
// pizza in cosul de cumparaturi
      $this->pizza[$pizza_id] += $quant;
}

function remove_pizza($pizza_id, $quant) {
// metoda folosita pentru a scoate o anumita cantitate de
// pizza din cosul de cumparaturi
      if ( $this->pizza[$pizza_id] > $quant ) {
              $this->pizza[$pizza_id] -= $quant;
              return true;
      }
      else {
         return false;
      }
}

function empty_basket() {
// metoda pentru a goli cosul de cumparaturi
      if($this->pizza)
            $this->pizza=NULL;

      $this->value=0;
}

function get_value() {
// metoda pentru a calcula valoarea cosului
     $this->value = 0;
     $aux = array();
     $aux = array_keys($this->pizza);

     $keys = implode(",", $aux);
     $result = getRows("pizza", "id_pizza", $keys);
     $aux = array();
```

```
    while ( $row = mysql_fetch_object($result) ) {
        array_push($aux,$row->PRET);
    }

    $i=0;
    foreach( $this->pizza as $val ) {
        $this->value += $val * $aux[$i++];
    }
}

function get_pizza(){
// metoda pentru a afisa produsele din cos
    if( $this->pizza ){
            foreach($this->pizza as $key => $val)
              echo "$key ";
            echo "<br />";
            foreach($this->pizza as $key => $val)
            echo "$val ";
    }
 }
} // final class cos
```

Pentru adăugarea produselor în coşul de cumpărături se foloseşte următorul cod:

```
$un_cos = &new cos(); //instantiem un nou cos
session_register("un_cos"); //inregistram variabila
// adaugam in cos
$_SESSION["un_cos"]->add_pizza($id_pizza, $cantitate);
```

iar pentru a goli coşul se foloseşte un apel de genul:

```
$_SESSION["un_cos"]->empty_cos();
```

Pentru a se transmite comanda efectiv am folosit funcţia:

```
function NewOrder($user, $data, $id_pizza, $cant,
                  $livrata, $platita) {
    global $link;

    $query = "INSERT INTO comenzi_a
(user,data_c,id_pizza,cant,livrata,platita) VALUES
('$user','$data',$id_pizza,$cant,'$livrata','$platita')";

    $result = mysql_query($query, $link) or
                  die("Error:" . mysql_error());

    return $user;
```


Modulul de administrare a sitului

Pentru a nu supraîncarca serverul MySQL, am decis ca acest modul să fie implementat cu ajutorul tehnologiei XML. Astfel, am creat fişierul `noutati.xml` care conţine toate noutăţile afişate în pagina principală a sitului.

Exemplul următor ilustrează structura documentului XML folosit:

```
<noutati>
<noutate id="6">
  <imp>3</imp>
  <data>2003/04/06</data>
  <text>E-Pizza lanseaza un nou concurs, astfel puteti fi
fericitul castigator al unei pizza super doar raspunzand
la chestionar (numai pentru utilizatorii inregistrati)</text>
  <titlu>Concurs!</titlu>
</noutate>
<noutate id="7">
  <imp>2</imp>
  <data>2003/06/05</data>
  <text>Numai saptamana aceasta puteti beneficia de oferta
speciala la pizza cu mere: cumparand 2 pizza cu mere mai
primiti una GRATUIT!</text>
  <titlu>Oferta speciala</titlu>
</noutate>
</noutati>
```

La procesarea acestui fişier va fi folosită biblioteca *XML22* prezentată în anexa D.

Având în vedere cerinţele funcţionale am divizat şi acest modul în două părţi: partea de administrator (care va permite adăugarea, modificarea şi ştergerea anumitor noutăţi, precum şi sortarea noutăţilor) şi partea de utilizator (care va afişa noutăţile aşa cum au fost ele sortate de către administrator).

Pentru a prelua fiecare noutate stocată de fişierul XML, vom procesa documentul XML şi vom căuta elementele şi atributele dorite, după care le vom afişa.

Pentru a sorta noutăţile în ordine crescătoare vom utiliza funcţia de comparaţie de mai jos:

```
function cmpd($a, $b) {
   if ($a["content"] == $b["content"]) return 0;
   return ($a["content"] > $b["content"]) ? -1 : 1;
}
```

Pentru sortarea în ordine descrescătoare se foloseşte:

```
function cmpa($a, $b) {
     if ($a["content"] == $b["content"]) return 0;
     return ($a["content"] > $b["content"]) ? 1 : -1;
}
```

Următorul exemplu ilustrează procesarea, sortarea după dată în ordine descrescătoare şi afişarea noutăţilor:

```
$doc = xml22_parse('noutati.xml'); // procesare XML
// preluarea tuturor elementelor
$res = xml22_get_all_of_name($doc, $field);
// sortare, folosind pt. comparatii functia cmpd()
uasort($res, "cmpd");
// ne intereseaza doar tag-uri <data>
if ($field == "data") {
     $radacina = xml22_get_parent($doc, $aux);
     // preluam atributul id...
     $id = xml22_get_attribute($aux, 'id');
     // ...si il afisam
     print $id;

     // preluam si continutul lui <data> si-l afisam
     $data = xml22_get_content($aux);
     print $data;

     // preluam gradul de importanta <imp>
     $imp = xml22_get_prev_sibling($doc, $aux);
     $imp = xml22_get_content($aux);
     print $imp;

     // preluam textul efectiv al noutatii
     $text = xml22_get_last_child($aux);
     $text = xml22_get_content($aux2);
     print $text;
}
```

Pentru a adăuga o noutate pe o anumită poziţie, trebuie mai întâi procesat documentul XML, apoi trebuie preluată poziţia dorită şi construit nodul XML corespunzător împreună cu toate atributele şi fii săi direcţi şi, în sfârşit, adăugat nodul creat în document.

Următorul exemplu ilustrează adăugarea unei noutăţi:

```
$doc = xml22_parse('noutati.xml');
$res = xml22_get_all_of_name($doc,'noutate');

// sorteaza documentul XML dupa id-ul noutatilor
usort($res,"cmp");

$id = xml22_get_attribute($res[$cautat],'id');
$last = xml22_get_by_id($doc,$id);
```



```
$id++;
$noutate = xml22_create_tag
              ('noutate','',$last, XML22_BYSIBLING);

$noutate = xml22_add_attribute($noutate,'id',$id);
$doc = xml22_add_sibling($doc,$noutate,$last);

// genereaza tag-ul <titlu> al noutatii create
$titlu = xml22_create_tag
              ('titlu', $titlu, $noutate, XML22_BYPARENT);
$doc = xml22_add_child($doc,$titlu,$noutate);

// creeaza tag-ul <text> al noutatii
$text = xml22_create_tag
              ('text', $text, $noutate, XML22_BYPARENT);
$doc = xml22_add_child($doc,$text,$noutate);

// creeaza tag-ul <data>
$data = xml22_create_tag
              ('data', $data, $noutate, XML22_BYPARENT);
$doc = xml22_add_child($doc,$data,$noutate);

// creeaza tag-ul <imp> al noutatii create
$importanta = xml22_create_tag
              ('imp', $imp, $noutate, XML22_BYPARENT);
$doc = xml22_add_child($doc,$imp,$noutate);
// scrie in fisierul noutati.xml
// documentul incluzand noutatea adaugata
$fd = fopen("noutati.xml","w+");
xml22_write_document($doc, $fd, XML22_BEAUTIFY);
fclose($fd);
```

Pentru a modifica o noutate se creează *tag*-ul candidat pentru modificare şi apoi se înlocuieşte *tag*-ul nou creat cu cel vechi.

De exemplu, următorul cod va modifica elementul `<imp>`:

```
$res = xml22_get_by_id($doc, $id[$i]);
$_imp = xml22_create_tag
            ('imp', $imp[$i], $res, XML22_BYPARENT);

$res = xml22_get_first_child($doc, $res);
$doc = xml22_replace_tag($doc, $res, $_imp);
```

Pentru a face posibilă vizualizarea în ordinea selectată de către administrator vom folosi funcţia *bool array_multisort ()* pusă la dispoziţie de PHP. Lăsăm scrierea funcţiei de vizualizare ca exerciţiu pentru cititor.

Modulul de generare a graficelor

Pentru a genera graficele vom folosi biblioteca *JpGraph* (prezentată în anexa E) după care vom culege datele din baza de date şi apoi, pe baza acestor informaţii, vom genera fişierele grafice finale (a se vedea şi cele expuse în cadrul capitolului 6).

De exemplu, pentru a genera graficul care reflectă comparaţia dintre numărul de comenzi autentificate şi numărul de comenzi neautentificate pe zile, am preluat din baza de date numărul de comenzi autentificate pe zile folosind următoarea interogare SQL:

```
$query = "SELECT DAYNAME(comenzi_a.DATA_C) AS ZIUA,
COUNT(comenzi_a.CANT) AS TOTAL FROM comenzi_a GROUP BY
ZIUA ORDER BY `ZIUA` DESC";
```

Pentru a prelua numărul de comenzi rapide vom scrie:

```
$query = "SELECT DAYNAME(comenzi_q.DATA_C) AS ZIUA,
COUNT(comenzi_q.CANT) AS TOTAL FROM comenzi_q GROUP BY
ZIUA ORDER BY `ZIUA` DESC";
```

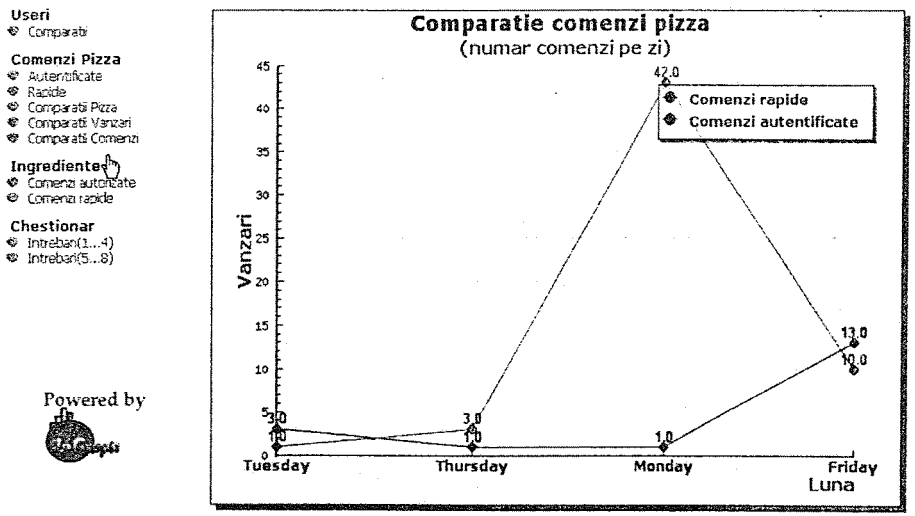
Pentru a genera graficul s-a recurs la următorul cod:

```
// preia date comenzi rapide
$result = getTop10ComenziLuni();
while ( $row = mysql_fetch_object($result) ) {
            array_push($xdata,$row->ZIUA);
            array_push($ydata1,$row->TOTAL);
}
// preia comenzi autentificate
$result = getTop10ComenziLuniI();
while ( $row = mysql_fetch_object($result) ) {
            array_push($ydata2,$row->TOTAL);
}

// instantiaza obiectul graph
// se va genera o imagine de 650x450 pixeli
$graph = new Graph(650, 450, "auto");
$graph->img->SetAntiAliasing();
$graph->SetScale("textlin");

// seteaza legenda
$graph->legend->Pos(0.05, 0.15);
$graph->legend->SetShadow('darkgray@0.5');
$graph->legend->SetFillColor('azure2@0.3');
$graph->legend->SetFont(FF_VERDANA, FS_BOLD, 10);
```



```
// seteaza culorile graficului
$graph->SetMarginColor('azure2');

// creeaza prima linie
$p1 = new LinePlot($ydata1);
$p1->SetColor("blue");
$p1->mark->SetType(MARK_FILLEDCIRCLE);
$p1->mark->SetFillColor("blue");
$p1->mark->SetWidth(3);

// se vor vizualiza valorile cu precizie de o zecimala
$p1->value->SetFormat('%0.01f');
$p1->value->SetFont(FF_ARIAL, FS_BOLD, 10);
$p1->value->SetColor("blue");
$p1->value->Show();
$p1->SetLegend('Comenzi rapide');

// creeaza a doua linie
$p2 = new LinePlot($ydata2);
$p2->SetColor("red");
$p2->mark->SetType(MARK_FILLEDCIRCLE);
$p2->mark->SetFillColor("red");
$p2->mark->SetWidth(3);

// valorile vor fi vizibile cu precizie de o zecimala
$p2->value->SetFormat('%0.01f');
$p2->value->SetFont(FF_ARIAL, FS_BOLD, 10);
$p2->value->SetColor("red");
$p2->value->Show();

// stabileste titlul legendei
$p2->SetLegend('Comenzi autentificate');

// adauga liniile graficului
$graph->Add($p1);
$graph->Add($p2);

// seteaza margini, titlu, subtitlu
$graph->img->SetMargin(40,20,20,40);
$graph->title->Set("Comparatie comenzi pizza ");
$graph->title->SetFont(FF_VERDANA, FS_BOLD, 14);
$graph->subtitle->Set("(numar comenzi pe zi)");
$graph->subtitle->SetFont(FF_VERDANA, FS_BOLD, 12);
```

```
// seteaza titlurile axelor
$graph->xaxis->title->Set("Luna");
$graph->yaxis->title->Set("Vanzari");

// seteaza fonturile axelor
$graph->yaxis->title->SetFont(FF_VERDANA, FS_BOLD, 12);
$graph->xaxis->title->SetFont(FF_VERDANA, FS_BOLD, 12);

// seteaza culorile axelor
$graph->yaxis->SetColor("black");
$graph->yaxis->SetWeight(2);
$graph->yaxis->scale->SetGrace(1);

// seteaza valorile coordonatei carteziene X
$graph->xaxis->SetTickLabels($xdata);
$graph->xaxis->SetFont(FF_VERDANA, FS_BOLD, 10);

// seteaza o umbra pentru grafic
$graph->SetShadow();

// seteaza o margine pentru imagine
$graph->img->SetMargin(60, 50, 50, 50);

// afiseaza graficul
$graph->Stroke();
```



Graficul generat cu ajutorul bibliotecii *JpGraph*

În codul prezentat mai sus am folosit funcţia *SetAntiAliasing()* – avizăm cititorul că această funcţie este mare consumatoare de memorie.

Imaginea ilustrează graficul generat de codul prezentat anterior.

Modulul de rezervări on-line

Având în vedere cele discutate mai sus, am împărţit acest modul în două submodule: o parte care se va ocupa de administrare şi o parte care e destinată clientului (adică vizitatorului care va dori să-şi rezerve o masă).

Partea de administrator

Această parte a aplicaţiei este bazată pe limbajul *JavaScript*, realizându-se o procesare a datelor pe partea client (direct în navigator) pentru că avem nevoie de actualizarea dinamică a interfeţei Web, adică a configuraţiei meselor care sunt rezervate.

Acest modul trebuie să îndeplinească două scopuri importante:

- să ofere administratorului posibilitatea de a muta mesele din restaurant (modificarea, deci, a unei configuraţii de mese);
- să ofere administratorului posibilitatea de a adăuga sau şterge mese.

Pentru a îndeplini aceste două aspecte am implementat o funcţionalitate pentru acţiunea *drag-and-drop*, folosind trei funcţii:

- *MD()* – funcţie care tratează evenimentul de apăsare a unui buton de *mouse*, fără a fi încă eliberat (*mouse down*);
- *MM()* – funcţie care tratează evenimentul de mutare a cursorului *mouse*-ului (*mouse move*);
- *MU()* – funcţie care tratează evenimentul de eliberare a unui buton (*mouse up*).

În funcţia *MD()* se captează obiectul de interfaţă pe care s-a executat *click*. Este comparat *id*-ul obiectului cu şirul „mese" – pentru a fi siguri că serviciul de *drag-and-drop* va fi disponibil numai pentru mesele care sunt afişate pe pagină şi nu pentru alte obiecte de interfaţă (cum ar fi coşul de gunoi utilizat la eliminarea rezervării unei mese sau alte butoane). În cazul în care obiectul vizat este o masă, se apelează funcţia *MM()*. Acest lucru îl realizăm în maniera descrisă în continuare.

Pentru început avem nevoie de câteva variabile globale definite astfel:

```
// variabilele globale folosite
var ob, old;
var mese_i = new String();
var splitMese = new Array();
```

Variabilele de mai sus au următorul rol:

- *ob, old* – sunt obiectele JavaScript asociate obiectelor de interfaţă. Variabila *ob* face referire la obiectul de interfaţă deasupra căruia a fost ridicat butonul *mouse*-ului, iar obiectul *old* referă obiectul peste care a fost apăsat butonul *mouse*-ului. Acest lucru este necesar de ştiut deoarece în momentul în care se efectuează un *drag-and-drop* trebuie verificate şi atributele „src" ale imaginilor

meselor asupra cărora se efectuează operaţia. După prima operaţie de *drag-and-drop*, se interschimbă două imagini („masa liberă" cu „masa ocupată", de exemplu), dar *id*-urile lor nu vor fi schimbate. Deci este necesar să ştim care este obiectul de unde a început operaţia de *drag-and-drop* şi unde se termină pentru a face posibilă şi interschimbarea *id*-urilor;

- *mese_i* şi *splitMese* sunt două variabile care fac legătura între scriptul PHP (executat pe server) şi programul JavaScript (rulat pe client). Astfel, în codul HTML al paginii Web se află inserat un câmp ascuns (*hidden*) care are rolul de a obţine de la scriptul PHP configuraţia meselor din restaurant. Acest lucru este obţinut de scriptul PHP prin interogarea tabelei *config_restaurant*, responsabilă cu stocarea datelor privitoare la configuraţia actuală a meselor de restaurant. Valoarea acestui câmp este preluată în variabila *mese_i*. În acest câmp numerele meselor sunt separate prin caracterul „|". Tabloul *splitMese* va conţine pe fiecare poziţie numărul unei mese.

Codul funcţiilor *MD()* şi *MM()* este prezentat mai jos:

```
function MD() {
// se trateaza evenimentul mouse down
     ob = event.srcElement;
     old = event.srcElement;
     if (ob.id.substring(0, 4) == "mese") {
            ob=document.getElementById("dragging");
            ob.style.visibility = "visible";
            MM();
     }
     else ob = 0;
     mese_i = document.forms["myform"].rezervari.value;
     splitMese = mese_i.split("|");
}

function MM() {
// se trateaza evenimentul mouse move
     if ( ( event.button == 1 ) && ob) {
     ob.style.pixelLef =
                    event.clientX+document.body.scrollLeft;
     ob.style.pixelTop=
                    event.clientY+document.body.scrollTop;
     }
}
```

În funcţia *MU()* se execută de fapt toate operaţiile pe care trebuie să le efectueze operaţia de *drag-and-drop*. Deoarece şi adăugarea de mese este implementată tot prin intermediul operaţiei de *drag-and-drop* şi pentru că în restaurant există un număr limitat de mese care pot fi aşezate, un prim caz care trebuie tratat de funcţia *MU()* este

verificarea, în cazul adăugării unei mese, a numărului de mese care se află deja plasate în restaurant. Dacă acest număr este egal cu cel din baza de date, adăugarea nu poate fi efectuată.

În cazul ştergerii de mese nu este necesară verificarea numărului de mese plasate deoarece în momentul în care nu mai există mese pe zona de plasament (reprezentând sala reală a restaurantului), datorită modalităţii de implementare a operaţiei de *drag-and-drop*, aceasta nu se va mai putea fi efectuată. Ştergerea de mese implică următoarele: din controlul de tip *hidden*, care conţine mesele care sunt disponibile la un moment dat în restaurant, este eliminată masa care a fost trimisă la coşul de gunoi.

Implementarea propusă are în vedere şi tratarea corespunzătoare a eventualelor evenimente neplăcute. În momentul în care este trasă imaginea (reprezentând o masă) folosită pentru a adăuga mese în restaurant peste coşul de gunoi, în cadrul scriptului trebuie să avem grijă să nu se şteargă din controlul *hidden* identificatorul unei mese care a fost efectiv plasată. Un alt caz tratat de funcţia *MU()* este validarea poziţiei unde se eliberează butonul *mouse*-ului. Acest lucru este necesar deoarece am putea muta o masă peste o altă masă plasată sau am putea să o mutăm pe ecran la o poziţie din afara zonei de plasare a meselor.

Înainte însă de a putea fi executată operaţia de *drag-and-drop*, programul JavaScript are nevoie de date care există în bazele de date deţinute de server (aranjamentul iniţial al meselor, numărul total de mese disponibile etc.). Pentru efectuarea operaţiei de *drag-and-drop*, trebuie să existe în prealabil o zonă (de tip *grid*) care să conţină imaginile meselor şi aranjamentul acestora în restaurant. *Grid*-ul este de fapt un tabel creat cu ajutorul unui script PHP, având la creare numărul de celule egal cu spaţiul maxim disponibil pentru mesele din cadrul restaurantului. Generarea configuraţiei iniţiale a meselor se face obţinând din baza de date un şir special, format din 0 şi 1. Aceste simboluri au următoarea semnificaţie:

0 – semnifică faptul că pe poziţia curentă nu există plasată nici o masă;
1 – semnifică faptul că pe poziţia respectivă există deja plasată o masă.

În cadrul acestui modul nu ne interesează dacă o masă este sau nu este ocupată. Ne interesează doar poziţia ei în restaurant. Tot în momentul creării tabelei, se verifică dacă în şirul extras din baza de date se află un 0 sau un 1. În funcţie de rezultatul verificării, se inserează în celula respectivă fie imaginea „floor.jpg", fie imaginea „table.jpg". În această iteraţie este format şi *id*-ul imaginilor. Acest *id* este format din cuvântul „liber" sau „mese", plus un număr unic pentru identificare. În momentul în care administratorul apasă pe butonul de salvare sunt trimise fişierului *modificari.php* toate schimbările configuraţiei de mese survenite în restaurant. Acest ultim script PHP este responsabil cu actualizarea bazei de date, pentru a reflecta situaţia actuală a restaurantului.

Partea de client

Acest modul va trebui să efectueze următoarele:

- să permită utilizatorului să vizualizeze situaţia meselor;
- să permită utilizatorului să rezerve o masă;

- să permită utilizatorului să anuleze o rezervare;
- să permită utilizatorului să vizualizeze detaliile legate de rezervarea sa.

Pentru a face o rezervare, am pus la dispoziţia utilizatorului două metode de abordare:

- prin intermediul operaţiei de *drag-and-drop*;
- prin intermediul meniurilor contextuale.

Modalitatea de efectuare a rezervărilor este similară cu cea prezentată mai sus. Se impun aproximativ aceleaşi restricţii ca în cazul modulului administrator. Astfel, în acest modul, au fost efectuate verificări pentru a împiedica un utilizator să rezerve o masă deja rezervată şi pentru a opri efectuarea operaţiei de *drag-and-drop* într-o zonă de ecran care este diferită de zona *grid*-ului. Aceste lucruri se realizează similar ca în cazul precedent.

Primul lucru pe care va trebui să-l realizeze acest modul este acela de a prezenta utilizatorului zona de dispunere a meselor din restaurant. În cazul de faţă ne interesează şi dacă o anumită masă este rezervată sau nu. Deci, într-o anumită celulă, va putea exista la un moment dat una dintre următoarele trei imagini:

- „floor.jpg" – semnifică faptul că în acest loc nu există nici o masă;
- „table.jpg" – semnifică faptul că o masă nu este rezervată;
- „table_r.jpg" – semnifică faptul că o masă este rezervată.

Pentru a cunoaşte starea unei mese (rezervată sau nerezervată), vom urma pentru început paşii de la modulul anterior. Aceasta constă în interogarea tabelei *config_restaurant* pentru a obţine în primul rând felul în care sunt dispuse mesele în restaurant. După acest pas, pentru fiecare masă va trebui realizată o interogare SQL asupra tabelei *mese* care conţine date despre o anumită masă. În funcţie de valoare câmpului *ocup*, în celula din *grid* corespunzătoare va fi afişată una dintre imaginile „table.jpg" sau „table_r.jpg". În cazul în care în această celulă nu se află nici o masă, va fi afişată imaginea „floor.jpg". Tot acum vor fi formate şi *id*-urile imaginilor folosite.

Implementarea operaţiei de *drag-and-drop* este similară celei prezentate în modulul administrator.

În continuare vom discuta despre cea de a doua posibilitate de a efectua o rezervare, cea prin intermediul meniurilor contextuale afişate la apăsarea butonului drept al *mouse*-ului.

O primă problemă în proiectarea şi implementarea meniului contextual apare datorită posibilităţii de realizare în paralel a operaţiei *drag-and-drop*. Această operaţie este declanşată de apăsarea unui buton al *mouse*-ului. Se observă că nu contează care este butonul de *mouse* apăsat, evenimentul se produce oricum. Din acest motiv apare un conflict cu implementarea operaţiei de meniu contextual, în momentul în care este apăsat butonul drept al *mouse*-ului, deoarece se va executa funcţia *MD()*. Pentru a rezolva această problemă, s-a implementat o funcţie JavaScript care va returna ce buton al *mouse*-ului a fost apăsat. Codul pentru această funcţie este prezentat mai jos:


```
function mouseDown(e) {
// rezolva evenimentul mouse down
     if (parseInt(navigator.appVersion) > 3) {
            var clickType=1;
            clickType=event.button;

            if (clickType==1) button_p = 0;
            if (clickType!=1) button_p = 1;
     }
     return true;
}

if ( parseInt(navigator.appVersion) > 3 ) {
     document.onmousedown = mouseDown;
}
```

În primul rând este verificată versiunea navigatorului deoarece aceasta trebuie să fie mai mare decât trei pentru ca această funcţie să fie executată corect. Pentru a vedea ce buton a fost apăsat a fost captat evenimentul pentru *mouse* şi a fost verificată variabila *clickType*.

După ce am implementat această funcţie, avem control asupra butoanelor pe care le apasă utilizatorul. În cadrul marcatorului <body> al documentului am inserat o condiţie de genul:

```
if ( buton_p == 0 )
  MD();
```

Acest lucru determină ca funcţia *MD()* să fie apelată numai în momentul în care este apăsat butonul stâng al *mouse*-ului. În caz contrar, va fi afişat meniul contextual. Pentru a realiza efectiv meniul contextual am folosit două funcţii: una care afişează şi una care ascunde meniul. Codul pentru aceste două funcţii este prezentat mai jos:

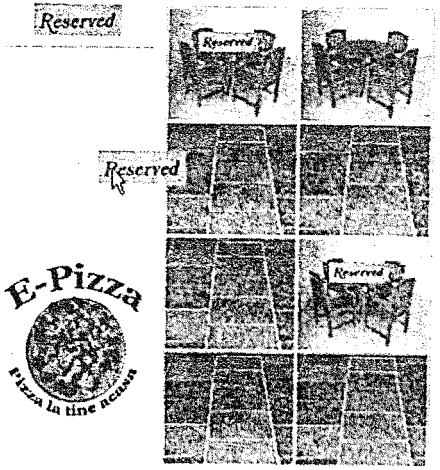
```
     var amMeniuContext = 0;

     function arataMeniu(evnt) {
          divu.style.left = evnt.clientX;
          divu.style.top = evnt.clientY;
          divu.style.visibility = 'visible';
          amMeniuContext = 1;
     }

     function ascundeMeniu() {
          divu.style.visibility = 'hidden';
          amMeniuContext = 0;
     }
```

Prin intermediul meniului contextual, utilizatorul va putea să-şi rezerve o masă sau să vadă detaliile legate de rezervarea lui.

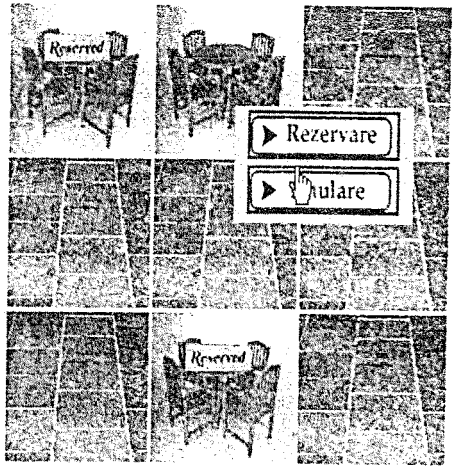
În momentul în care utilizatorul încearcă să rezerve o masă deja rezervată, aplicaţia verifică imaginea care a produs evenimentul de „*click* dreapta" sau evenimentul de *drag-and-drop*. Dacă imaginea este „table_r.jpg" este generat un mesaj de avertisment şi rezervarea nu are loc.

Un alt inconvenient este reprezentat de încercarea utilizatorului de a anula rezervarea unei mese, rezervare care nu a fost realizată de către acesta. Pentru a preveni această situaţie, aplicaţia reţine numele persoanei autentificate, iar în baza de date fiecărei mese îi este asociat un şir de caractere care va păstra numele persoanelor care rezervă masa respectivă. Astfel, în momentul în care utilizatorul selectează „Anulare" din meniul contextual asociat imaginii, se va executa o verificare asupra identităţii utilizatorului. Dacă acesta este persoana care într-adevăr a efectuat rezervarea, acţiunea lui este dusă la bun sfîrşit, altfel se va genera un mesaj de avertisment.

Următoarele imagini ilustrează procesul de *drag-and-drop* precum şi meniul contextual:



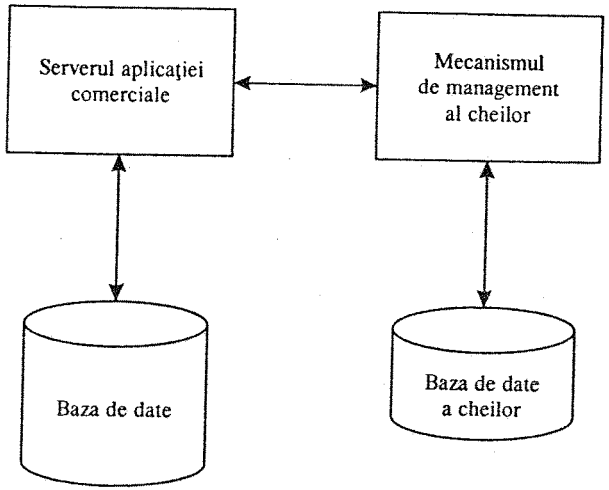
Efectuarea operaţiei *drag-and-drop* în afara zonei de mese



Activarea meniului contextual

Modulul de plată electronică

În dezvoltarea acestui modul o prima cerinţă a fost proiectarea unei arhitecturi software care să se integreze uşor cu aplicaţiile de tip *e-bussiness* deja existente. Astfel, a fost proiectat un sistem în care există un proces extern oferind managementul cheilor de criptare (generare, criptare, decriptare). Elementele criptografice ale aplicaţiei sunt deja implementate în software-ul conţinut în distribuţia serverului Web (a se consulta şi capitolele 2 şi 3). Diagrama următoare oferă o privire de ansamblu asupra topologiei sistemului:

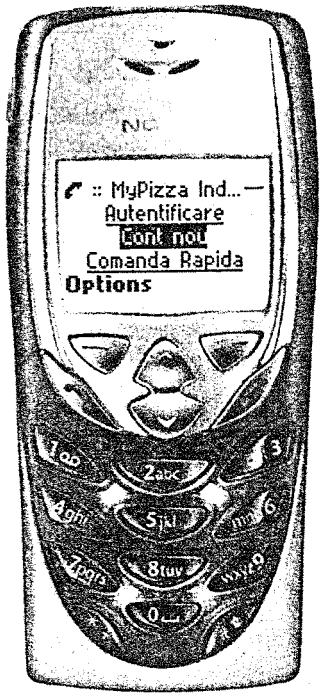


Interacţiunea între aplicaţie şi modulul de management al cheilor

Pentru realizarea mecanismului de management al cheilor s-a dezvoltat clasa *EPay* al cărei şablon îl prezentăm în cele ce urmează:

```
public class EPay {

// Genereaza si pastreaza in baza de date o cheie
// pentru o perioada definita de timp
public void keyGen (int period, String passPhrase);

// Pentru o cheie de tip Base64 regaseste perioada de
// gratie si cheia publica
public String keyRetrieve (int period, String passPhrase);

// Invecheste o cheie pe o perioda de timp si sterge cheia
// din baza de date definitiv
public void keyDeprecate (int period);

// Cripteaza un sir de caractere numit clearTarget
// folosind o cheie privata folosind
// un algoritm de tip DES, 3DES sau AES
public String encrypt (String key, String clearTarget,
String algorithm);

// Decripteaza o cheie folosind un cifru
// si o cheie publica
// folosind un algoritm de tip DES, 3DES sau AES
```

```
public String decrypt (String key, String cipherText,
String algorithm);
}
```



Prima pagină a sitului accesat via un dispozitiv WAP

Modelul de comandare prin WAP

S-a recurs la limbajul *WML*, descris în anexa F a cărţii, pentru generarea paginilor care vor putea fi vizualizate cu ajutorul unor clienţi compatibili WAP (de exemplu, Ericsson ori Nokia) sau cu ajutorul unor emulatoare (*i.e.*, *M3Gate* şi *Wireless Companion*) – a se vizita şi situl www.wapforum.org.

Pentru a genera prima pagină am folosit următorul cod WML:

```
<?xml version="1.0" ?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
          "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<!-- Pagina principala -->
 <card id="index" title=":: MyPizza Index ::">
    <p align="center">
```



```
      <strong>
         Bine ati venit pe situl WAP al pizzeriei noastre
      </strong>
    </p>
    <p align="center">
    <a href="#login">Autentificare</a><br/>
    <a href="signin.php">Cont nou</a><br/>
    <a href="#quickC">Comanda Rapida</a><br/>
    <a href="contact.php">Contact</a>
    </p>
 </card>

<!-- Comanda rapida -->
 <card id="quickC" title="Comanda Rapida">
  <p align="center">
     Introduceti datele pentru o comanda rapida.
     Campurile * sunt obligatorii.</p>
  <p><img src="new-1.wbmp" alt="LOGO"/></p>
  <do type="prev" label="Inapoi"><prev/></do>
 </card>

<!-- Autentificare -->
 <card id="login" title="_Autentificare_">
    <p align="center"><strong>
      Introduceti Utilizatorul si Parola</strong></p>
    <p>USER:
       <input type="text" name="USER" format="*M"/></p>
    <p>PASS:
       <input type="password" name="PASS"
              format="*M"/></p>
    <p>
       <anchor> Autentifica!
          <go method="post" href="auth.php">
             <postfield name="authInfo"
                        value="$(USER)@$(PASS)"/>
          </go>
       </anchor>
    </p>
    <do type="prev" label="Inapoi"><prev/></do>
    </card>
  </wml>
```

Pentru a afişa meniul avem următorul cod PHP cu care am generat tabele în limbaj WML.

Vizualizarea sortimentelor de pizza

```
echo "<wml>\n";
echo "<card id=\"Meniu\" title=\"Meniul\">\n";
echo "<p> Meniul Zilei !!</p>\n";
// obtinem sortimentele de pizza
$query = "SELECT * FROM `pizza` ORDER BY NUME_P ASC";
$result = mysql_query($query,$link);
// generam tabelul WML
print "<p><table columns=\"1\" align=\"c\">";
while ($row = mysql_fetch_object($result)){
        print "\n<tr>\n"  ;
        print "\t<td>";
        print "<anchor>$row->NUME_P";
        print "<go  method=\"post\"
                    href=\"detalii.php\">";
        print "<postfield name=\"id_pizza\"
                    value=\"$row->ID_PIZZA\"/>";
        print "<postfield name=\"PHPSESSID\"
                    value=\"" . session_id() . "\"/>";
        print "</go>";
        print "</anchor>";
        print "\t</td>\n";
```



```
        print "</tr>\n" ;
        $counter++;
}
print "</table></p>";
print "<do type=\"prev\" label=\"Inapoi\"><prev/></do>";
echo "</card>";
echo "</wml>";
```

Avizăm cititorul asupra faptului că pentru ca navigatorul WAP să proceseze corect codul WML trebuie să expediem conţinut aparţinând tipului MIME `text/vnd.wap.wml`. Pentru aceasta, înainte de trimiterea marcajelor WML, va trebui scrisă o linie de genul:

```
header ("Content-type: text/vnd.wap.xml");
```

Pentru unii clienţi WAP trebuie adăugate alte două linii, care să indice navigatorului să nu salveze documentul WML în memoria *cache*:

```
header ("Cache-control: no-cache, must-revalidate");
header ("Pragma: no-cache");
```

Detaliile despre o pizza sunt afişate ca în imaginea alăturată, iar codul pentru aceasta pagină este următorul:

Vizualizarea detaliilor privitoare la un sortiment selectat

```
echo "<wml>";
echo "<card id=\"Detalii\" title=\"Detalii\">\n";
// preluam detaliile din baza de date
$query = "select * from pizza where id_pizza=$id_pizza";
if (!$result = mysql_query($query,$link))
     print(mysql_error());
```

```
while($row = mysql_fetch_object($result))
      {
     print"<p><b>Pizza: </b><u>$row->NUME_P</u></p>";
     print"<p><b>Marime: </b>$row->MARIME cm</p>";
     print"<p><b>Greutate: </b>$row->GREUTATE g</p>";
     print"<p><b>Pret: </b>$row->PRET Lei</p>";
     print"<p><b>Ingrediente: </b>";
     getIngred($row->INGREDIENTE);
     print"</p>";
     }
print "<p align=\"center\">____________</p>";
print "<p>Cantitatea:";
// setam valoarea initiala pentru cantitatea comandata
print "<setvar name=\"cantitate\" value=\"0\"/>";
// generam un camp de interactiune cu utilizatorul
print "<input type=\"text\" name=\"cantitate\" value=\"0\"
        size=\"3\" maxlength=\"2\" format=\"*N\"/></p>";
print "<p>";
print "<anchor> Comanda ";
// actiunea de adaugare in cos
// cu trimiterea cantitatii si tipului de pizza
print "<go method=\"get\" href=\"adauga_c.php\">";
print "<postfield name=\"cantitate\"
                    value=\"$(cantitate)\"/>";
print "<postfield name=\"tipp\" value=\"$id_pizza\"/>";
print "<postfield name=\"PHPSESSID\"
                    value=\"" . session_id() . "\"/>";
print "</go>";
print "</anchor>";
print "</p>";
print "<do type=\"prev\" label=\"Inapoi\"><prev/></do>";
echo "</card>";
echo "</wml>";
```

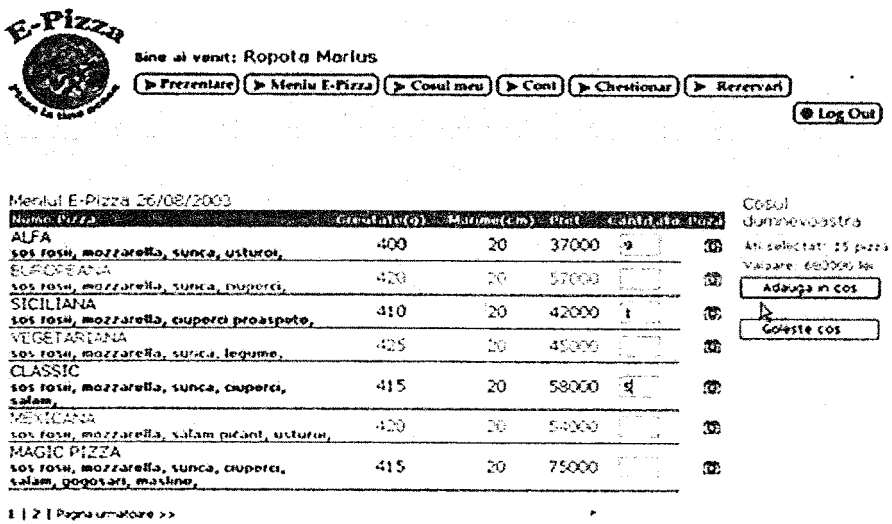
2. Ghid de utilizare

Acestă secţiune prezintă modul de funcţionare a aplicaţiei *E-Pizza*.

Prezentarea aplicaţiei pentru clienţi

La accesarea paginii principale (`index.php`) utilizatorul va putea vizualiza noutăţile apărute sau efectua o comandă rapidă. De asemenea, va avea posibilitatea să-şi creeze un cont nou sau – în cazul în care este deja înregistrat – să se autentifice (*login*). Pentru utilizatorii care şi-au uitat parola, există posibilitatea de a primi o parolă nouă.

Pagina pricipală pentru utilizatorii înregistraţi are interfaţa următoare:



Posibilitatea de a vizualiza oferta de pizza
în cazul utilizatorilor autentificaţi

Pentru a efectua o comandă se va completa cantitatea dorită dintr-un anumit sortiment de pizza şi se va apăsa butonul **Adaugă în coş**. Câmpul **cantitate** se va completa doar cu valori numerice din intervalul [1,99]. Dacă se doreşte anularea comenzii se va apăsa butonul **Goleşte coş**.

Pentru a vizualiza detalii despre un anumit sortiment de pizza se va apăsa pe imaginea aferentă fiecărui sortiment de pizza. Meniul poate fi sortat după greutate, mărime sau preţ.

Utilizarea coşului de cumpărături

Dacă nu s-a selectat nici o cantitate de pizza spre a fi comandată, coşul de cumpărături va afişa un mesaj care ne redirectează spre meniu.

În cazul în care a fost selectată o anumită cantitate de pizza, coşul de cumpărături va afişa această cantitate, preţul unitar, preţul pe sortiment de pizza şi valoarea totală a comenzii. Pentru a modifica cantitatea de pizza selectată se va modifica câmpul **cantitate** după care se va apăsa butonul **Actualizează**.

În cazul când se doreşte adăugarea unui alt produs se va apăsa butonul **Adaugă alt produs**. Dacă se doreşte anularea comenzii se va apăsa butonul **Renunţă**. Pentru a comanda efectiv se va apăsa pe butonul **Comandă**.

Nume Pizza	Pret Unitar	Cantitate	Pret Total
ALFA	37000	18	666000
EUROPEANA	57000	1	57000
SICILIANA	42000	1	42000
CLASSIC	58000	10	580000

Total: 1345000

Actualizeaza | Adauga alt produs | Comanda | Renunta

Captură-ecran a coşului de cumpărături

Prezentarea aplicaţiei pentru administrator

După conectare, administatorul are următoarele opţiuni la dispoziţie:

- *Noutăţi*

La accesarea legăturii *Noutăţi* se va deschide pagina pentru editarea, adăugarea şi ştergerea noutăţilor, precum se observă în imaginea următoare:

Adauga Noutati

Importanta
Data (yyyy/mm/dd)
Titlu
Text
Adauga

Modifica Noutati

Id	Data(yyyy/mm/dd)	Importanta	Titlu
1	2003/04/06	1	E-Pizza premiat
2	2003/04/06	3	Situl E-Pizza
3	2003/05/30	1	Rezultate concurs 2003/05/28
4	2003/06/20	3	Tombola
5	2003/01/06	1	Mare Concurs
6	2003/04/06	3	Foarte IMPORTANT
7	2003/06/05	2	Pizza Forest
8	2003/05/11	3	Castiguri

modifica sau sterge campurile selectate

Administrarea noutăţilor

- *Meniu*

La accesarea acestui *link* se va deschide pagina care conţine meniul restaurantului, astfel încât administratorul poate edita sortimentele de pizza, având posibilitatea

sortării conform diverselor criterii (identificator, nume pizza, greutate, mărime, preţ, vegetariană).

Imaginea următoare prezintă meniul administratorului:

Tabel ingrediente
(id-ul ingredient ->> nume ingredient)

1->>sos rosii	2->>mozzarella	3->>sunca	4->>ciuperci	5->>ciuperci proaspete
6->>legume	7->>salam	8->>salam picant	9->>sos picant	10->>porumb
11->>ton	12->>lamaie	13->>gogosari	14->>ananas	15->>file merluciu
16->>costita	17->>camat	18->>ou	19->>ceapa	20->>pastrama pui
21->>telemea	22->>masline	23->>fructe de mare	24->>scoici	25->>usturoi

Meniul E-Pizza 26/08/2003

ID	Nume Pizza	Greutate	Marime	Pret	Vegetariana
1	ALFA	400	20	37000	NU
2	EUROPEANA	420	20	57000	NU
3	SICILIANA	410	20	42000	NU
4	VEGETARIANA	425	20	45000	DA
5	CLASSIC	415	20	58000	NU
6	MEXICANA	420	20	54000	NU
7	MAGIC PIZZA	415	20	75000	NU
8	MONTANA	420	20	60000	NU
9	FRUCTE DE MARE	410	20	89000	NU
12	FOREST	500	30	47000	NU

modifica sau sterge campurile selectate

Nume Pizza
Greutate
Marime
Pret
Vegetariana
Ingrediente
Poza
Reteta
Adauga

1 | 2 | Pagina urmatoare >>

Editarea ingredientelor

- *Comenzi*

Administratorul are următoarele opţiuni în cazul în care a deschis pagina de comenzi: poate vizualiza toate comenzile, pentru a vedea comenzile livrate şi neplătite, ori poate consulta arhiva comenzilor:

Comenzi Autentificate
- Astazi
- Livrate si neplatite
- Arhiva

Comenzi Rapide
- Astazi
- Arhiva

ID	User	Data	Detalii
1	Nichifor Vali	2003-03-27 13:44:55	
2	IONUT ION	2003-04-26 23:25:25	
3	Nechita Cristian	2003-04-27 13:43:19	
4	Marin Marin	2003-04-27 13:43:42	
5	Ion ION	2003-05-26 23:23:56	
6	Nichifor Vali	2003-05-27 13:44:55	
7	Ropota Marius	2003-06-20 23:22:45	
8	Ropota Marius	2003-06-21 23:22:45	
9	Rotberg Iavi	2003-06-22 23:23:08	
10	Rotberg Iavi	2003-06-22 23:23:30	

Sterge campurile selectate sau

Cauta o inregistrare

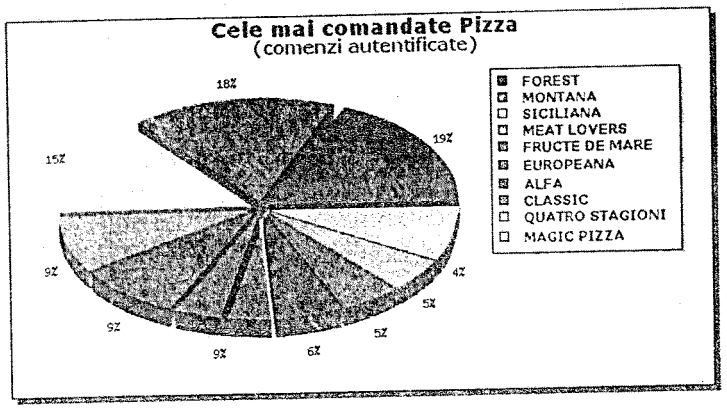
1 | 2 | Pagina urmatoare >>

Vizualizarea comenzilor

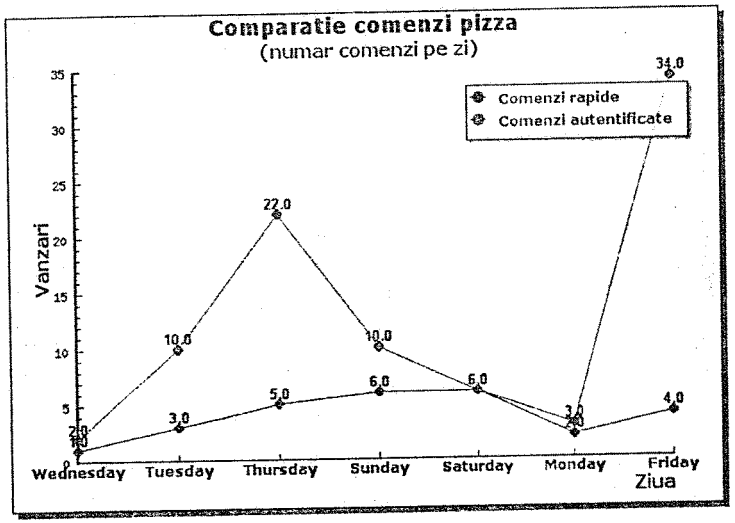
• *Top10*

Această secţiune afişează diferite clasamente privitoare la activitatea sitului *E-Pizza*. Administatorul are la dispoziţie următoarele legături:

- *Autentificate* – aici se pot observa care sunt cele mai comandate pizza prin intermediul comenzilor autorizate.
- *Rapide* – se oferă o imagine care evidenţiază cele mai comandate pizza prin sistemul de comenzi rapide.
- *Comparaţii pizza* – se oferă o comparaţie a celor mai vândute pizza pe luni.
- *Comparaţii vânzări* – se pune la dispoziţie o imagine a vânzărilor pe fiecare lună.
- *Comparaţii comenzi* – se pot observa care sunt cele mai solicitate zile.

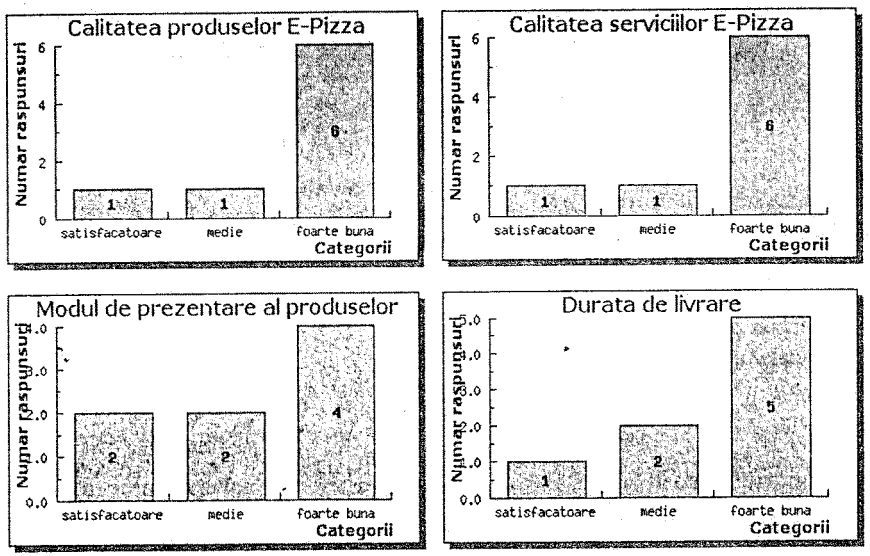


Vizualizarea situaţiei celor mai comandate tipuri de pizza



Comparaţia grafică între tipurile de comenzi efectuate

- *Comenzi autorizate* – se oferă imaginea celor mai consumate ingrediente în cadrul comenzilor autentificate.
- *Comenzi rapide* – se afişează cele mai consumate ingrediente în cazul comezilor rapide.
- *Întrebări (1...4)* – afişează graficele cu răspunsurile utilizatorilor pentru primele patru întrebări ale chestionarului disponibil *on-line* pe sit. Întrebările se referă, printre altele, la calitatea produselor şi serviciilor oferite, la modul de prezentare şi la durata de livrare a sortimentelor comandate *on-line*.



Vizualizarea grafică a răspunsurilor utilizatorilor

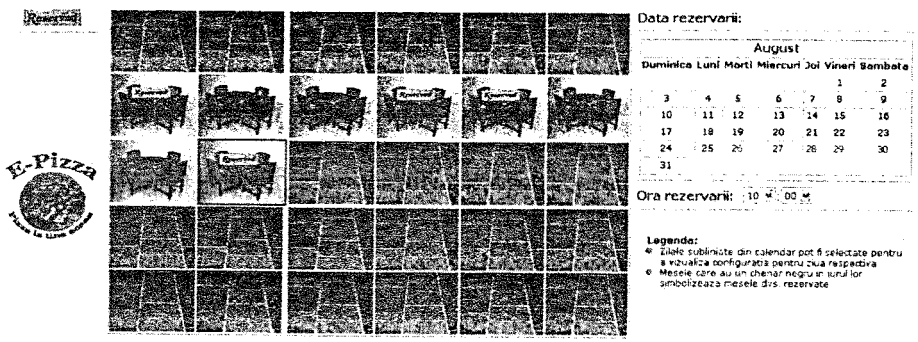
- *Întrebări (5...8)* – generează graficele cu răspunsurile utilizatorilor pentru următoarele întrebări ale chestionarului *on-line*.

În ceea ce priveşte rezervările de mese, administratorul are la dispoziţie următoarele opţiuni:

• *Rezervări (client)*

Pentru a rezerva o masă, sunt puse la dispoziţie două modalităţi: folosind operaţia de *drag-and-drop* sau prin intermediul unui meniu contextual (a se vedea secţiunea 1.4). Pentru a rezerva o masă folosind prima manieră, se efectuează operaţia de *drag* asupra imaginii etichetate *Rezervat*. Ţinând butonul *mouse*-ului apăsat, aceasta este trasă până deasupra mesei care se doreşte a fi rezervată, după care butonul e eliberat (*drop*). Pentru a folosi meniul contextual, se efectuează *click* cu butonul drept pe o masă.

O rezervare poate fi făcută şi cu câteva zile înainte. Pentru a realiza acest lucru, se efectuează *click* pe data la care se vrea a fi realizată rezervarea dorită. Pentru a confirma rezervarea se foloseşte butonul **Confirmă rezervarea**.



Modul de rezervare a meselor (client)

• *Rezervări (administrator)*

Administratorul poate să genereze configuraţii pentru mesele restaurantului. Aceste configuraţii pot fi aleatorii sau pot fi create după dorinţă, modificându-se aranjamentul meselor, prin adăugare sau ştergere de mese de pe suprafaţa de lucru (*grid*).

Interfaţa prezintă trei zone importante:

- meniurile;
- *grid*-ul meselor;
- acţiunile disponibile.

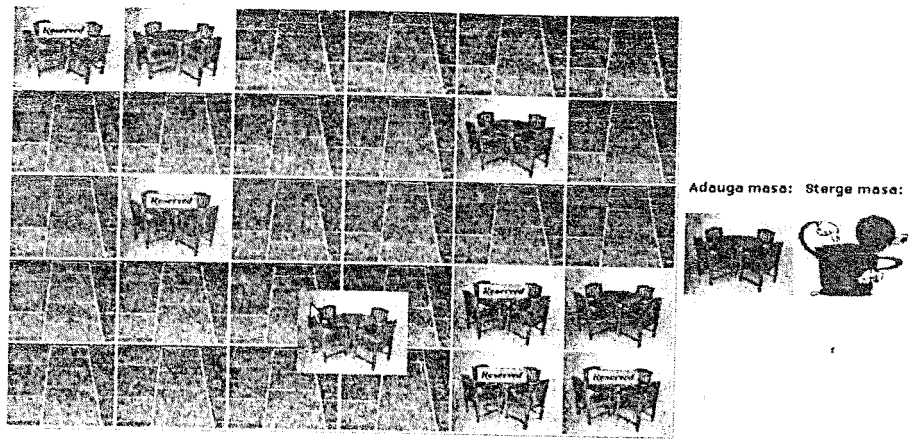
Meniurile sunt situate în partea superioară a ferestrei navigatorului. Ele asigură acces rapid la ziua dorită din săptămână, oferind şi modalităţi de a şterge toate mesele, de a genera configuraţii aleatorii şi de a modifica numărul de mese. Pentru a selecta o altă zi a săptămânii, administratorul va trebui să efectueze un *click* stâng pe numele zilei pe care vrea să o vizualizeze.

Există posibilitatea ca, la un moment dat, administratorul să dorească să şteargă toate mesele de pe *grid*. Acest lucru se poate efectua şi executând operaţia de *drag-and-drop* a imaginilor meselor deasupra coşului de gunoi, dar dacă există multe mese această acţiune este anevoioasă. Astfel, pentru a şterge toate mesele, administratorul va activa opţiunea **Şterge mesele**.

Pentru a genera o configuraţie aleatorie – prin aceasta înţelegându-se faptul că mesele sunt poziţionate pe *grid* la întâmplare –, administratorul va trebui să folosească opţiunea **Generează o configuraţie aleatorie**. După generarea unei astfel de configuraţii, administratorul va avea posibilitatea să o modifice în mod tradiţional.

O ultimă opţiune din partea de meniuri este **Modifică numărul de mese**. Prin intermediul acesteia, administratorul are posibilitatea de a modifica numărul de mese care sunt disponibile în restaurant într-o anumită zi.

Cele două acţiuni sunt reprezentate de fapt prin două imagini situate în partea stângă a ferestrei având semnificaţia: adăugare de mese şi, respectiv, ştergere de mese. Şi în acest caz există posibilitatea de a efectua *drag-and-drop*.



Adăugarea unei mese prin operaţia *drag-and-drop*

3. Concluzii

În cadrul acestui capitol am descris o soluţie – bazată pe diferite tehnologii Web actuale – de implementare a unui sit de *e-commerce*, în particular varianta WWW a unei pizzerii. Aplicaţia prezentată, *E-Pizza*, oferă o interfaţă clasică şi una orientată WAP pentru realizarea de comenzi de sortimente de pizza. De asemenea, există posibilitatea ca un client să rezerve o masă în restaurantul (real) al pizzeriei, să vizualizeze noutăţile zilei ori să completeze un formular on-line, răspunzând la diverse întrebări referitoare la activitatea sitului ori pizzeriei. *E-Pizza* pune la dispoziţie un set de instrumente de administrare *on-line* a sitului şi de editare a listei de produse şi ingrediente. De asemenea, administratorul are posibilitatea de a vizualiza comod diverse grafice, informându-se despre evoluţia vânzărilor, numărul de comenzi efectuate, tipurile de sortimente cele mai solicitate etc.

Implementarea aplicaţiei a recurs la tandemul clasic PHP + MySQL, dar şi la folosirea unor biblioteci PHP *open-source* pentru procesarea documentelor XML (a se vedea anexa D) ori pentru generarea dinamică de grafice (a se consulta anexa E). Pentru interfaţa cu dispozitivele WAP s-au utilizat scripturi PHP care au generat dinamic marcatori WML (vezi şi anexa F). În ceea ce priveşte realizarea de rezervări de mese, în vederea asigurării unei cât mai mari flexibilităţi în utilizare, s-a optat pentru o soluţie bazată pe JavaScript în implementarea operaţiei de *drag-and-drop* şi a meniurilor contextuale.

Cititorii interesaţi pot încerca, pornind de la cele expuse mai sus, să dezvolte în întregime situl *E-Pizza*, recurgând eventual şi la alte soluţii de implementare sugerate de celelalte capitole ale cărţii.

Comentarii bibliografice

Problematica atinsă în capitolul de faţă este complementară celor abordate de capitolele 2 şi 3, aplicaţia prezentată fiind una de comerţ electronic (a se vedea referinţele bibliografice de la precedentele capitole). Noutatea constă în utilizarea documentelor XML pentru stocarea diverselor informaţii, în locul bazelor de date. Cititorul poate avea o imagine de ansamblu asupra limbajului XML consultând cărţile S. Buraga, *Tehnologii Web*, Matrix Rom, Bucureşti, 2001 şi S. Buraga, V. Tarhon-Onu, Ş. Tanasă, *Programare Web în bash şi Perl*, Polirom, Iaşi, 2002. De asemenea, se mai pot vizita siturile:

- XML.org: **www.xml.org**
- XML Software: **www.xmlsoftware.com**
- XML Times: **www.xmltimes.com**
- XML References, Resources and Demos: **www.cen.com/ng-html/xml/xml-refs.html**
- ZVON: **www.zvon.org**

În ceea ce priveşte programarea JavaScript, recomandăm consultarea specificaţiei oficiale disponibile pe situl Netscape: **http://developer.netscape.com/library/documentation/index.html**. Se mai pot vizita şi adresele: **http://www.ecma.ch**, **http://www.dlib.org**, **http://javascript.internet.com** sau **http://www.dynamicdrive.com**.

Drept resurse bibliografice pentru lumea WAP şi WML indicăm:

- WAP Forum: **www.wapforum.org**
- WAP Links: **www.waplinks.com**
- WAP.com: **www.wap.com**

Unelte de dezvoltare şi un tutorial WML sunt disponibile pe CD-ul care însoţeşte cartea S. Buraga, *Proiectarea siturilor Web*, Polirom, Iaşi, 2002.

Trebuie amintite şi adresele Web de la care se pot prelua bibliotecile folosite de aplicaţia *E-Pizza*:

- Biblioteca *XML22*: **http://www.ister.org/code/xml22/**
- Biblioteca *JpGraph*: **http://www.aditus.nu/jpgraph/**

CAPITOLUL 5

Geometrie... virtuală (cu VRML, SVG şi PHP)

Alexandrina Orzan, alexo@infoiasi.ro

Acest capitol descrie aplicaţia *GraphX3D*, un sit Web educaţional care construieşte reprezentarea grafică a unor curbe şi suprafeţe 3D. Se prezintă problematica abordată şi actuala soluţie imaginată, utilizându-se SVG şi X3D pentru redarea graficii Web. Se urmăresc apoi etapele de creare a arhitecturii sitului *GraphX3D* şi modul de implementare a structurii obţinute. Capitolul se încheie cu prezentarea rezultatului final, prin capturi-ecran şi un scurt manual de utilizare.

1. Web 2D şi Web 3D

În această secţiune vom descrie ideea care stă la baza implementării sitului *GraphX3D* şi vom prezenta două modalităţi de redare a graficii Web 2D şi 3D, utilizate în aplicaţie.

1.1. Problema „concretă" şi soluţia ei

Formularea problemei

Ideea de la care am pornit a fost că informatica ar putea uşura considerabil studiul unor domenii aride şi poate dificil de înţeles (pentru unii) din învăţământul universitar şi preuniversitar. Asta pentru că, utilizând calculatorul, cunoştinţele se pot preda într-un mod mai intuitiv şi mai atractiv.

Astfel, se pot realiza pachete-soft (în particular aplicaţii Web) care să fie destinate înţelegerii facile de către elev sau student a conceptelor prezentate. De exemplu, în predarea matematicii ar fi utile modelări de grafice de funcţii 2D şi 3D (un astfel de soft se poate utiliza de elevi pe toată perioada studiului gimnazial şi liceal), precum şi modelarea obiectelor tridimensionale, cu intersecţii de planuri şi drepte în spaţiu.

Soluţie

Urmărind punerea în practică a ideii de mai sus, am implementat un sit Web care oferă celor interesaţi un mediu interactiv pentru practicarea, învăţarea şi vizualizarea conceptelor fundamentale pentru curbe şi suprafeţe 3D. Acest sit se adresează, în principal, studenţilor care urmează cursuri de geometrie computaţională, fiind, de altfel, bazat pe cursul de Geometrie Computaţională (urmat la Facultatea de Informatică a Universităţii „Al.I. Cuza" din Iaşi).

Aplicaţia, numită *GraphX3D*, permite scrierea la intrare a unor ecuaţii parametrice sau a unor puncte de control pentru curbe/suprafeţe Bézier şi oferă drept rezultat graficul 3D al obiectului.

Problema care se pune este: cum se realizează reprezentările obiectelor 3D, când Web-ul are caracter 2D? Răspuns: cu ajutorul limbajului *VRML* (*Virtual Reality Modeling Language*). Ce înseamnă „Virtual Reality" şi cum arată limbajul care modelează realitatea virtuală vom explica în subsecţiunea următoare.

Ca totul să fie cât mai atractiv cu putinţă, utilizatorii au posibilitatea de a-şi defini propriile preferinţe în ceea ce priveşte aspectul sitului, adică pot schimba culoarea fundalului, a textului şi a graficii. Acest lucru este posibil pentru că grafica întregului sit este creată în *SVG* (*Scalable Vector Language*), limbaj despre care vom „povesti" mai multe în secţiunea 1.3.

1.2. Realitate virtuală (VR) şi VRML

Ce înseamnă realitate virtuală?

Realitatea virtuală (VR – *virtual reality*) reprezintă un mediu generat de calculator, care se adaptează la mişcările utilizatorului. De exemplu, atunci când utilizatorul îşi ridică privirea pentru a vedea, să zicem, un cer generat de calculator, cerul e acolo, iar atunci când utilizatorul se uită în jos, poate să vadă pământul „virtual" generat de calculator.

Esenţa realităţii virtuale este constituită de mediile 3D create de calculator. Aceste medii, numite „lumi", reprezintă imitaţii ale lumii reale sau medii abstracte prin care se poate naviga şi cu care se poate interacţiona.

O lume virtuală poate fi explorată de utilizatori prin intermediul unor echipamente precum o cască menită a acoperi ochii (*head-mounted display*) şi mănuşi cu senzori (*data-glove*). Acestea permit utilizatorului să se plimbe prin lumea virtuală şi să atingă obiecte.

Totuşi, cea mai răspândită formă de realitate virtuală utilizată acum este „desktop VR". În aceste sisteme, lumile virtuale rulează pe calculatorul utilizatorului, pot fi vizualizate pe un monitor standard şi permit navigarea cu ajutorul *mouse*-ului şi al tastaturii. Astfel de sisteme pot fi distribuite uşor pe Web şi au nevoie doar de un *viewer* pentru a putea fi vizualizate.

Formatul recunoscut pentru crearea şi vizualizarea de lumi virtuale pe Web este VRML.

Ce este VRML?

VRML (*Virtual Reality Modeling Language*) a fost dezvoltat de Consorţiul Web3D şi a fost destinat utilizării pe Internet, fiind văzut ca o extensie 3D a World-Wide Web. VRML este un format 3D pentru Web popular pentru că e relativ uşor de utilizat, este standardizat şi produce fişiere reduse ca dimensiuni.

VRML s-a lansat public la data de 3 aprilie 1995 şi specificaţia VRML versiunea 1.0 a urmat în luna mai a aceluiaşi an. Pentru că VRML este un format relativ simplu, construit pe un standard bine definit, un număr de instrumente pentru modelat VRML şi programe de convertire automată a altor formate 3D în VRML au fost create destul de repede. În 1996 se dă publicităţii specificaţia versiunii 2.0, iar în 1997 apare o specificaţie apropiată de VRML 2.0, numită VRML97. În prezent este propusă spre standardizare cea de-a treia versiune a limbajului: *X3D*. Limbajul X3D este compatibil cu VRML 2.0, dar sintaxa folosită pentru descrierea scenelor este XML, deoarece este sintaxa care predomină, la momentul actual, în cadrul Web-ului.

VRML 2.0 permite descrierea de ierarhii de obiecte simple, cum ar fi cuburi, cilindri şi sfere. Forme mai complexe pot fi de asemenea definite, la fel ca materialele din care e formată suprafaţa, textura suprafeţei şi date *LOD* (*Level Of Detail*). Obiectele pot fi legate de alte obiecte prin URL-uri.

În limbajul VRML pot să se realizeze reprezentări statice şi animate de obiecte şi pot exista hiper-legături către alte elemente multimedia, cum ar fi sunetele, filmele, imaginile şi textul.

VRML defineşte şi senzori care simplifică operaţiile efectuate cu ajutorul *mouse*-ului (de exemplu, *drag-and-drop*). Senzorii permit detectarea coliziunii şi apropierea utilizatorului şi interacţiuni ale utilizatorului, precum *click* şi *drag*, iar timpul poate fi măsurat şi exprimat.

VRML permite încorporarea de programe de tip script (concepute de exemplu în JavaScript ori Java), făcând astfel posibilă programarea comportării obiectelor din lumea virtuală. De exemplu, poate fi folosit un script pentru a schimba modul de iluminare atunci când utilizatorul se plimbă prin spaţiul virtual.

Cum pot fi vizualizate lumile VRML?

Pentru a putea naviga printr-o lume VRML este necesar un vizualizator (*viewer*) de VRML. Interpretoarele (vizualizatoarele) VRML sunt disponibile (de cele mai multe ori gratuit) pentru diferite platforme, la fel cum există şi diverse instrumente capabile să creeze şi să manipuleze fişiere VRML.

Una dintre companiile care oferă soft pentru VRML gratuit (inclusiv vizualizator) este Parallel Graphics, cu localizarea pe Web la adresa **www.parallelgraphics.com**. O altă adresă, la care puteţi găsi un vizualizator gratuit pentru sistemul Linux, este **http://www.crc.ca/FreeWRL/**.

Ce este necesar pentru a crea lumi VRML?

Fişierele VRML pot fi scrise în orice editor de text. Există şi editoare specializate, care oferă trăsături precum *AutoComplete* sau crearea automată a ierarhiei de obiecte.

De asemenea, există şi o serie de unelte care permit crearea de lumi VRML în mod grafic (gen Photoshop pentru obiecte 3D). Un astfel de produs este oferit gratuit de Parallel Graphics.

1.3. Grafica vectorială şi SVG

În această subsecţiune vom prezenta pe scurt limbajul *SVG* (*Scalable Graphics Language*). Datorită faptului că o trăsătură importantă a acestui limbaj este aceea de a permite realizarea de grafică vectorială, vom vorbi mai întâi despre aceasta.

Introducere în grafica vectorială

Grafica vectorială descrie imaginile folosind linii şi curbe, nu pixeli, ca formatele clasice de tip raster (*bitmap*). Aceste linii şi curbe, alături de informaţii despre culoare şi poziţie, formează **vectorii**.

Atunci când se editează un „grafic vectorial", se modifică proprietăţile liniilor şi curbelor care-i descriu forma. Astfel, se pot schimba poziţia, mărimea, forma şi culoarea unui „grafic vectorial" fără a se altera calitatea imaginii. Acest mod de reprezentare determină **independenţa de rezoluţie** a graficii vectoriale. Ea poate fi văzută pe monitoare de rezoluţii diferite fără a-şi pierde din calitate, şi aceasta pentru că imaginea este desenată utilizând acelaşi set de vectori, dar redimensionaţi după cum este nevoie.

Adaptarea fără probleme la orice tip de ecran este principalul avantaj al utilizării graficii vectoriale în loc de metoda clasică a *bitmap*-urilor. *Bitmap*-urile descriu imaginea utilizând colecţii de pixeli aranjaţi într-o matrice şi sunt dependente de rezoluţie, pentru că datele compunând imaginea sunt fixate pentru o anumită dimensiune (cea a matricei).

Ce este SVG?

SVG (*Scalable Vector Graphics*) este un limbaj utilizat pentru descrierea de imagini bidimensionale în *XML* (*Extensible Markup Language*) dezvoltat de Consorţiul World-Wide Web (W3C). SVG dă posibilitatea dezvoltatorilor Web de a crea grafică interactivă şi personalizată de înaltă calitate, generată dinamic cu date în timp real (graficele se pot modifica automat, în funcţie de informaţiile de care depind).

SVG permite folosirea a trei tipuri de obiecte grafice: forme descrise prin grafica vectorială (obiecte conţinând linii drepte şi curbe), imagini şi text.

Obiectele SVG pot fi interactive şi dinamice. Animaţiile pot fi definite şi declanşate fie declarativ (prin includerea elementelor de animare SVG în fişier), fie utilizând fişiere de tip script. Compatibilitatea limbajului SVG cu alte standarde Web conduce la utilizarea facilă de caracteristici – de exemplu, scripturile – care pot acţiona simultan asupra elementelor SVG şi XHTML în cadrul unei aceleiaşi pagini Web.

Datorită trăsăturilor precum adaptarea la orice rezoluţie şi formatul text, formatul SVG a fost adoptat de Comunitatea Dispozitivelor Mobile. Companii importante, ca Nokia, Ericsson şi OpenWave, s-au implicat în realizarea unor specificaţii pentru telefoane mobile şi dispozitive *PDA* (*Personal Digital Assistant*). Rezultatul a fost *Mobile SVG*.

SVG Mobile Profiles (SVG Basic şi SVG Tiny) au fost standardizate în ianuarie 2003, iar SVG Tiny este declarat de Comunitatea Dispozitivelor Mobile ca un format obligatoriu pentru transmiterea de mesaje multimedia. Tot atunci SVG versiunea 1.1 a devenit recomandare a Consorţiului Web.

Caracteristici ale limbajului

SVG a fost creat pentru a face ca imaginile Web să se potrivească oricărui ecran, inclusiv ecranelor de telefoane mobile. Dezvoltat cu contribuţia unor importante companii software şi hardware – ca Adobe, Corel, Apple şi Microsoft –, formatul SVG este un format text, ceea ce conduce la încărcarea mai rapidă a graficii Web. În plus, acest format text permite motoarelor de căutare să localizeze informaţiile scrise în cadrul imaginilor, lucru care nu este posibil în cazul *bitmap*-urilor, pentru că textul şi numerele sunt transformate în pixeli.

Specificaţia SVG oferă alternative pentru multe dintre funcţionalităţile puse la dispoziţie de Flash şi încorporează multe altele. Structura SVG se bazează pe crearea de elemente prin grafica vectorială şi include posibilitatea de a defini animaţii bazate pe timp pentru fiecare element, la fel ca Flash. Mai mult, SVG permite scrierea de scripturi de procesare a documentelor XML – via *DOM* (*Document Object Model*) – în limbaje ca JavaScript, ECMAScript sau oricare altul suportat de vizualizatorul SVG. În plus, formatul text permite parcurgerea conţinutului documentelor SVG, nemaifiind astfel nevoie, ca în cazul Flash-ului, de adăugarea de informaţii suplimentare pentru roboţii Web.

Elemente grafice de bază – ca dreptunghiul, cercul, linia, elipsa, poligonul, linia frântă şi textul – sunt definite în specificaţia SVG. Utilizând schema SVG, se pot defini fonturi proprii şi se poate chiar modela forma textului după o curbă. Imagini *bitmap*, în formatele PNG, GIF, JPG, pot fi inserate în documentele SVG cu elementul `image`.

Proprietăţile elementelor SVG pot fi definite prin atributul `style`, dar pot fi precizate şi folosind standarde Web, cum ar fi *Cascading Style Sheets* (*CSS*) şi *Extensible Style Sheet Language* (*XSL*), pentru ca grafica realizată să poată fi uşor de modificat.

Toate atributele pot fi animate cu ajutorul scripturilor sau folosind elementul (*tag*-ul) `animate`. Animaţia este implementată în SVG utilizând *SMIL* (*Synchronized Multimedia Integration Language*), de asemenea o tehnologie dezvoltată de către Consorţiul Web. Folosirea sintaxei SMIL permite includerea de documente SVG într-o prezentare SMIL multimedia, fiind astfel posibilă suprapunerea de animaţie realizată cu grafică vectorială peste elemente audio şi video.

Implementarea SVG ca format bazat pe XML oferă utilizatorilor avantajul de a crea grafică SVG din alte date XML. Transformarea datelor XML în SVG prin XSL

permite trecerea simplă de la date în format text la reprezentări grafice, ceea ce face ca posibilităţile de folosire a SVG-ului să fie practic nelimitate. Un exemplu ar fi transformarea, în timp real, a unui document XML conţinând o statistică a populaţiei într-un grafic animat care arată în manieră intuitivă evoluţia demografică. Mai mult, dacă datele din XML se modifică, atunci se schimbă automat şi graficul generat prin SVG.

Limbajul SVG poate fi folosit şi într-o manieră similară Flash-ului, pentru a crea meniuri de navigare prin situri Web sau chiar pentru a genera situri întregi. Utilizarea graficii SVG pentru meniuri Web elimină necesitatea unui număr mare de imagini. De exemplu, un buton pentru navigare care necesită, în mod normal, două imagini *bitmap*, poate fi înlocuit uşor de un singur fişier SVG, care schimbă dinamic atributele elementelor sale. De asemenea, SVG este interactiv, oferă suport pentru indicarea de hiper-legături şi poate fi inclus *inline* în fişierele HTML.

Pentru că SVG este un format XML, toate tehnologiile XML pot fi folosite de dezvoltatorii SVG, inclusiv standarde ca DOM şi XSLT sau biblioteci de procesare XML, precum cele din JSP, ASP şi PHP. De asemenea, SVG poate fi folosit alături de alte documente XML. Astfel, SVG se poate integra în fişiere XHTML sau poate fi combinat cu MathML pentru a crea reprezentări grafice pentru formule matematice.

Datorită simplităţii schemei XML, fişierele SVG se pot scrie fără mari probleme în orice editor de text, dar există o varietate de vizualizatoare şi editoare pentru documente SVG. Cel mai des utilizat *viewer* în momentul de faţă este *plug-in*-ul creat de Adobe, care permite vizualizarea de fişiere SVG din orice *browser* şi care este disponibil gratuit. Acesta se poate găsi la adresa **http://www.adobe.com/svg/viewer/install/main.html**.

De asemenea, Adobe a inclus suport pentru editarea de documente SVG în aplicaţia Illustrator 10. Alte companii, precum JASC, Corel, Sun şi IBM, au inclus suport SVG în produsele lor.

2. GraphX3D – arhitectura conceptuală

În cele ce urmează vom descrie paşii făcuţi pentru realizarea structurii aplicaţiei GraphX3D. Conceperea arhitecturii acestui sit este împărţită în trei etape: design-ul sitului, design-ul interfeţei şi design-ul paginii. La fiecare etapă, am urmărit respectarea standardelor de design Web recunoscute, astfel încât rezultatul final să fie un sit bine structurat şi având o interfaţă grafică uşor de folosit de către utilizatori.

2.1. Design-ul sitului

Design-ul unui sit determină cadrul organizatoric al acestuia. În timpul acestei etape se determină ce aşteaptă utilizatorul de la noi, ce dorim să spunem, şi cum aranjăm conţinutul sitului pentru ca acesta să fie uşor de parcurs. Experţii afirmă că, deşi grafica paginilor este cea mai rapid observată de utilizator, organizarea întregului sit are cel mai mare impact asupra acestuia.

Primul pas în conceperea oricărui sit este definirea clară a scopurilor. Fără obiective precise de urmărit, proiectul se va împotmoli sau va produce rezultate

diferite de cele aşteptate. **Scopul** acestei aplicaţii este realizarea unui mediu interactiv pentru practicarea, învăţarea şi vizualizarea conceptelor fundamentale pentru curbe şi suprafeţe 3D.

Următorul pas este **identificarea potenţialilor utilizatori** ai sitului Web, pentru a putea structura design-ul sitului în concordanţă cu nevoile şi dorinţele lor. *GraphX3D* se adrezează tuturor celor interesaţi de geometria în spaţiu, fie ei novici sau experţi în navigarea pe Web. Ca urmare, design-ul va trebui să combine informaţiile ajutătoare pentru începători cu accesul rapid la date de interes pentru cunoscători.

O dată stabilit despre ce este vorba în sit şi cui îi este adresat acesta, trebuie să se hotărască exact **ce va conţine situl**. Scopul acestei etape este de a aduna toate informaţiile necesare pentru crearea structurii sitului.

Situl *GraphX3D* trebuie să conţină, în principal, informaţii despre diferite obiecte din geometria 3D şi pagini care să permită construirea reprezentării grafice a tipurilor de obiecte prezentate.

Deoarece ecuaţiile parametrice permit definirea matematică a tuturor curbelor şi suprafeţelor regulate, am hotărât ca o modalitate de definire a corpurilor geometrice prezente în sit să fie cea prin ecuaţiile lor parametrice. Deşi acest mod de definire este foarte util în geometria clasică, el nu este foarte des folosit în geometria computaţională, pentru că nu oferă o imagine intuitivă a reprezentării grafice. Din acest motiv, am ales includerea în sit şi a curbelor şi suprafeţelor Bézier, cele mai utilizate modalităţi de reprezentare a obiectelor în geometria computaţională.

Pentru toate tipurile de obiecte geometrice, sunt necesare informaţii care să explice modalitatea de definire a lor. Situl *GraphX3D* va cuprinde, aşadar, şi scurte prezentări teoretice ale obiectelor modelate.

Alături de informaţiile precizate mai sus, am găsit utilă includerea în sit a unei biblioteci de obiecte 3D. Aceasta are scopul de a arăta capacităţile sitului şi de sublinia legătura dintre reprezentarea matematică şi cea grafică a obiectelor 3D.

GraphX3D îşi propune, de asemenea, să ofere utilizatorilor posibilitatea de a modela aspectul sitului. Pentru memorarea utilizatorilor şi a preferinţelor acestora este nevoie de informaţii de conectare (*login*); acestea vor trebui şi ele incluse în sit.

O regulă în design-ul siturilor Web este că întotdeauna trebuie incluse informaţii despre sit şi despre autorii lui. Acest tip de informaţii va fi, prin urmare, prezent şi în situl *GraphX3D*.

În plus, pentru utilizatorii neexperimentaţi în navigarea pe Web, am decis includerea unei secţiuni de ajutor care să răspundă eventualelor întrebări. Mai mult, este oferită posibilitatea de a contacta autorul pentru nelămuriri sau nemulţumiri în legătură cu aplicaţia.

Organizarea informaţiei

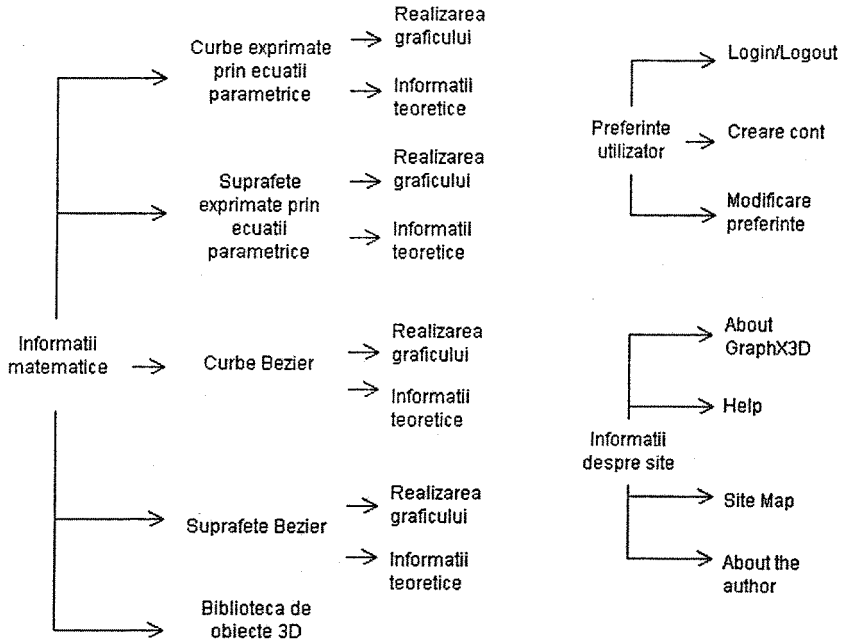
O dată stabilite informaţiile ce vor incluse în sit, acestea trebuie organizate în unităţi logice care vor constitui baza pentru structura sitului. Monitorizarea modului în care oamenii caută şi utilizează informaţia de pe Web sugerează că datele separate în unităţi mai mici sunt mai funcţionale şi mai uşor de utilizat. Numărul de unităţi recomandat de psihologi este între patru şi şapte, pentru fiecare nivel de informaţie.

În organizarea datelor dintr-un sit trebuie parcurse patru etape:

- împărţirea conţinutului în unităţi logice;
- stabilirea unei ierarhii între aceste unităţi;
- structurarea de relaţii bazate pe ierarhia anterior stabilită;
- construirea unui sit care respectă îndeaproape această structurare a informaţiei.

Respectarea organizării ierarhice precizate mai sus este o necesitate în design-ul sitului. Toate siturile sunt organizate ierarhic, pornind de la pagina principală şi „coborând" spre submeniuri din ce în ce mai specializate. O dispunere logică a informaţiei permite utilizatorilor să îşi dea seama repede unde pot găsi anumite date, iar metodele de organizare consistente fac uşoară tranziţia de la paginile familiare la cele nevizitate încă. O structură care nu este nici logică, nici previzibilă, va fi foarte greu de utilizat de către cei care vizitează situl.

Conform paşilor descrişi mai sus, am obţinut pentru aplicaţia *GraphX3D* următoarea structurare ierarhică a informaţiei:



Împărţirea în unităţi logice a informaţiei

Structura sitului

Siturile Web sunt construite în jurul unor teme structurale de bază. Aceste arhitecturi fundamentale guvernează modul de navigare prin sit şi creează modelul mental al utilizatorului despre cum este organizată informaţia. Pentru a construi un sit Web se pot folosi trei tipuri de structuri: secvenţă, ierarhie şi de tip *web* („pânză de păianjen").

Organizarea secvenţială este potrivită pentru situri în care utilizatorul trebuie să parcurgă un set de informaţii dinainte fixat şi singurele legături sunt cele care susţin navigarea liniară (de tip „înainte" sau „înapoi").

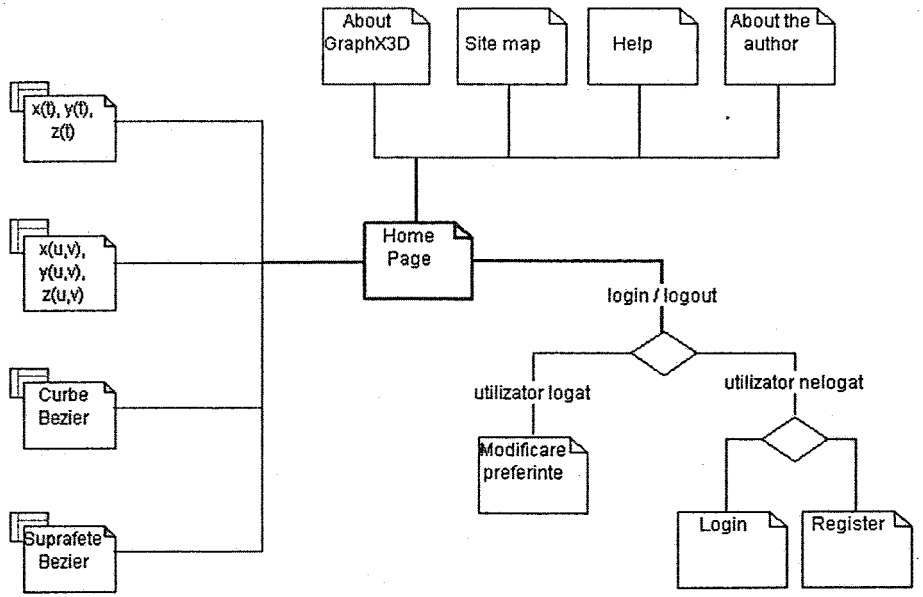
Organizarea ierarhică este cea mai des folosită, deoarece este potrivită pentru organizarea în jurul unei singure pagini principale. De asemenea, acest tip de organizare impune disciplină în organizarea materialului şi este foarte uşor de înţeles de către utilizator.

Organizarea de tip *web* are ca scop imitarea gândirii asociative şi a curgerii libere de idei. Dispunerea legăturilor şi a informaţiei nu este supusă nici unor restricţii şi de aceea poate crea deseori confuzii în rândul utilizatorilor. Acest mod de organizare poate fi folosit pentru situri Web având ca ţintă utilizatori experimentaţi sau cu studii superioare, care vor să aprofundeze un domeniu, nu să înţeleagă noţiunile de bază.

Pentru aplicaţia *GraphX3D* am hotărât utilizarea, în principal, a organizării ierarhice, deoarece este cea mai potrivită pentru un sit cu o gamă largă de tipuri de posibili utilizatori. Acolo unde a fost necesară impunerea unei anumite ordini de parcurgere a paginilor, am folosit structuri secvenţiale.

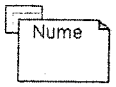
Utilizând împărţirea în unităţi logice a informaţiei şi temele structurale decise, am realizat o schemă arhitecturală a sitului. Aceasta este reprezentarea vizuală a structurii sitului şi constă în diagrame care arată cum sunt grupate elementele sitului şi care sunt legăturile dintre ele.

Schema sitului *GraphX3D* este:

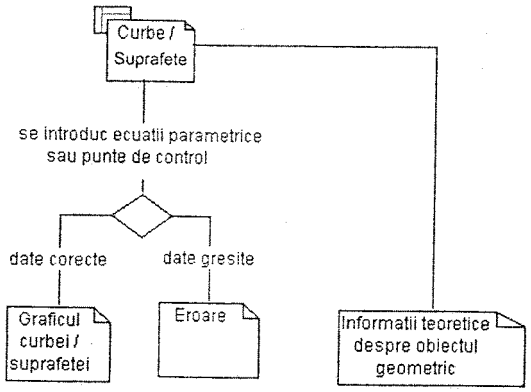


Schema arhitecturală a sitului *GraphX3D*

unde elementele de forma:



reprezintă structuri de genul:



Structură aparţinând schemei arhitecturale a sitului

2.2. Design-ul interfeţei

Interfaţa grafică cuprinde metaforele de interacţiune, imaginile şi conceptele utilizate pentru a reda funcţiuni şi sensuri pe un ecran de calculator. Ea include, de asemenea, caracteristicile vizuale ale tuturor elementelor de grafică care formează imaginea de ansamblu a paginilor Web. Grafica Web nu este folosită doar pentru a da viaţă paginilor Web – ea este esenţială pentru experienţa utilizatorului cu un sit Web. Design-ul grafic al documentelor interactive nu poate fi separat de design-ul interfeţei.

Din cele spuse mai sus rezultă că design-ul interfeţei este foarte important şi că trebuie, ca şi structura sitului, să se supună unor reguli stricte.

În aplicaţia *GraphX3D* am urmărit folosirea strategiilor de design universal recomandate pentru a crea o interfaţă plăcută, dar şi utilă şi uşor de folosit.

Pagini de sine stătătoare

Paginile Web diferă de cele dintr-o carte sau alte documente într-un aspect important: ele pot fi accesate, prin intermediul hiper-legăturilor, fără nici un preambul. Din acest motiv, paginile Web trebuie să fie mai independente decât paginile unei cărţi.

Cea mai bună strategie de design, în acest caz, este precizarea câtorva elemente de bază în toate paginile Web. Aceste elemente sunt:

- un titlu bine ales, care să informeze cu privire la conţinutul paginii (important pentru motoarele de căutare sau în cazul în care utilizatorul reţine adresa paginii într-un *bookmark*);
- identitatea creatorului (autor sau instituţie);
- data creării sau a ultimei actualizări;
- cel puţin o legătură la pagina principală sau la harta sitului.

În aplicaţia *GraphX3D* am urmărit îndeaproape aceste cerinţe, toate elementele de mai sus fiind prezente. Autorul şi data creării sitului sunt menţionate la sfârşitul fiecărei pagini, în *footer*, iar sigla *GraphX3D*, prezentă în *header*, are ataşată o legătură la pagina principală.

Definirea navigării în sit

Sistemul de navigare printr-un sit este foarte important. El determină modul de utilizare al sitului, hotărăşte cum se ajunge dintr-o locaţie a sitului în alta şi previne „pierderea" utilizatorului în „hăţişul" paginilor Web.

Utilizatorul trebuie să se poată întotdeauna întoarce uşor la pagina principală şi la alte puncte importante din sit (numărul acestora ar trebui să fie, de preferinţă, între patru şi şapte). Legăturile către aceste pagini trebuie să fie prezente, mereu în aceleaşi locaţii, pe toate paginile sitului Web. Design-ul asociat acestor legături de bază va crea o identitate grafică a paginilor care va asigura utilizatorul că se află încă în acelaşi sit.

GraphX3D păstrează acelaşi *header* (cap de pagină) pe fiecare pagină. Acesta constă într-o bară de legături ce asigură navigarea prin tot situl şi din sigla *GraphX3D*, care este, în acelaşi timp, legătură către pagina de start.

Această organizare este utilă din mai multe puncte de vedere. În primul rând, previne existenţa paginilor fără întoarcere. În al doilea rând, oferă acces în maxim doi paşi la toate informaţiile din sit (studiile au arătat că utilizatorii preferă meniurile dense, cu acces direct la informaţie, meniurilor simple, dar cu mai multe submeniuri).

Design-ul graficii

Utilizatorul nu este, în general, impresionat de o grafică încărcată şi complicată. Mai mult, *bitmap*-uri uriaşe care se încarcă greu nu vor produce decât frustrare. În general, grafica Web trebuie să fie simplă şi consecventă pe întreg situl.

În aplicaţia *GraphX3D* am încercat să realizăm o grafică plăcută, fără a încărca inutil paginile. Cele mai multe elemente de design sunt concentrate în antet, care este elementul-cheie pentru navigarea în întregul sit şi care trebuie să fie în mod clar separat de restul paginii.

Grafica sitului *GraphX3D* este realizată în întregime în SVG. Acest fapt oferă posibilitatea creării de grafică interactivă, atrăgătoare pentru utilizator, şi rezolvă problema încărcării greoaie a imaginilor (SVG este format text, uşor de transferat).

2.3. Design-ul paginii

Organizarea spaţială a graficii şi a textului pe o pagină Web poate atrage utilizatorii prin impactul grafic, poate să le dirijeze atenţia şi să organizeze potrivit importanţei informaţia pe care aceştia o văd. Un bun design al paginii face interacţiunea utilizatorilor cu situl nostru Web mai plăcută şi mai eficientă.

Ierarhie vizuală

Cea mai importantă sarcină a design-ului grafic este să creeze o ierarhie vizuală consecventă în care elementele importante sunt scoase în evidenţă, iar conţinutul este organizat logic şi predictibil.

Pentru a ghida vizual vizitatorul, design-ul grafic foloseşte dispunerea în pagină a elementelor şi ilustraţia. Cititorii înregistrează întâi culorile şi formele, apoi informaţiile prezente în elementele grafice, şi abia după aceea încep să citească cuvinte individuale şi fraze, de la stânga la dreapta, de sus în jos (în ţările cu acest tip de scriere).

Design-ul paginii trebuie să urmărească modul de parcurgere şi înregistrare a informaţiei de către ochiul uman. Din acest motiv, antetul sitului *GraphX3D*, prezent în toate paginile şi conţinând cele mai importante legături, este plasat în partea de sus a ecranului (cea care este citită prima). Pentru că este un element de *focus*, antetul este adus în prim-plan prin utilizarea de culori în contrast cu fundalul. Elementele grafice sunt şi ele folosite, pentru a sublinia legăturile cele mai importante, adică cele despre realizarea graficelor 3D. La fel, având în vedere stilul de parcurgere a informaţiei de la stânga la dreapta, în colţul din stânga sus este plasată sigla sitului, cu *link* spre pagina principală.

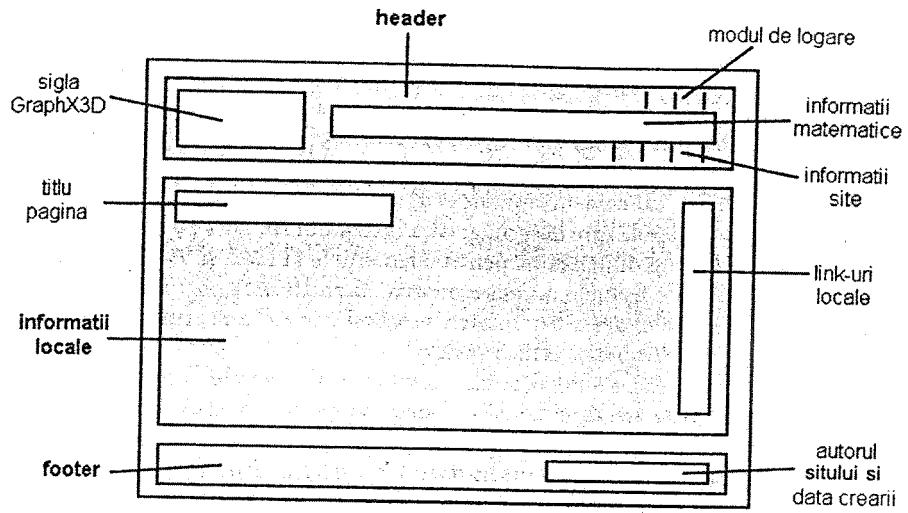
Consecvenţă

După cum am precizat şi în secţiunea dedicată structurii sitului, păstrarea unui aspect asemănător pe toate paginile acestuia este foarte importantă. Aceeaşi modalitate de dispunere a informaţiei pe pagină face uşoară orientarea utilizatorului şi creează impresia de tot unitar a sitului.

De aceea, în *GraphX3D* s-a urmărit păstrarea unei teme grafice pe întreg situl. Culorile sunt mereu aceleaşi şi toate imaginile sunt realizate într-un singur stil.

În plus, pentru a organiza logic şi consecvent paginile din care se compune aplicaţia de faţă am utilizat un *grid* (reţea, grilaj). Acesta determină unde anume pe pagină vor apărea, în mod regulat, diferite tipuri de informaţii şi ilustraţii. Practic, *grid*-ul este un şablon pentru toate paginile Web din sit; după ce acest *grid* este stabilit, în momentul implementării nu ne va mai rămâne de făcut nimic altceva decât să completăm ochiurile reţelei cu informaţiile specifice fiecărei pagini.

Organizarea paginilor după acest *grid* poate fi văzută în figura următoare:



Schema unei pagini din situl *GraphX3D*

Dimensiunea paginii

Dimensiunea paginii este un aspect ce nu trebuie neglijat în construcţia unui sit. Dacă paginile sitului au lăţimea mai mare decât cea a unui ecran de calculator obişnuit (de 17 sau 19 inci), atunci majoritatea utilizatorilor nu vor putea vedea întreaga pagină şi vor trebui să realizeze o derulare pe orizontală şi verticală. Dacă derularea verticală a conţinutului este acceptabilă, cea orizontală nu este tolerată de utilizatori.

Soluţia general acceptată pentru această problemă este restrângerea elementelor grafice pentru a încăpea pe un ecran cu rezoluţie de 800x600 pixeli (cel mai des întâlnit în mediile academice şi de afaceri). *GraphX3D* rezolvă această problemă altfel. Aplicaţia utilizează grafică realizată în SVG, deci grafică vectorială ce se poate adapta oricărei rezoluţii fără a-şi pierde din calitate.

3. GraphX3D – implementare

În acestă secţiune vom descrie modul de implementare a aplicaţiei *GraphX3D*, urmărind structura definită în secţiunea anterioară. Mai întâi vom prezenta organizarea fişierelor din aplicaţia de faţă, iar apoi vom preciza limbajele utilizate şi vom sublinia unele aspecte din scrierea codului în fiecare dintre aceste limbaje.

3.1. Organizarea fişierelor din sit

Situl *GraphX3D* este construit urmărind schema arhitecturală descrisă în secţiunea anterioară. Structura conceptuală este reflectată în modalitatea de împărţire în directoare

şi subdirectoare a fişierelor. Astfel, în directorul *GraphX3D* se află trei subdirectoare, `info site`, `info mate` şi `login`, alături de alte câteva fişiere necesare.

Fişierele aflate în *GraphX3D* sunt: `index.php`, `header.svg`, `footer.svg`, `body.svg`, `style.css`. Fişierul `index.php` este pagina principală a aplicaţiei. Acest script creează dinamic un document HTML care conţine cadre (*frame*-uri). În partea superioară se va încărca `header.svg`, iar în partea inferioară `footer.svg`. Restul zonei de afişare va fi dedicat conţinutului fişierului `body.svg`. Fişierele `header.svg` şi `footer.svg` sunt fişierele care descriu grafica întregului sit, iar `style.css` fixează atribute precum culoare şi dimensiune pentru elementele HTML şi SVG din acest sit.

În `login` se află toate fişierele necesare pentru autentificarea (*login*) utilizatorului şi modificarea preferinţelor acestuia. Tot aici se află şi `users.sql`, fişier care creează baza de date necesară memorării utilizatorilor.

În directorul `info site` sunt fişierele cu informaţii generale despre sit şi autor. Fiecare tip de informaţie are asociate două fişiere, unul *.php* şi unul *.xml*. În fişierul *.xml* se află informaţia propriu-zisă, organizată astfel încât să poată fi uşor de înţeles şi modificat de către un eventual administrator al sitului. Fişierul *.php* transformă construcţiile XML în pagini Web – a se vedea anexa C.

Directorul `info mate` conţine toate informaţiile matematice din sit. Acesta este cel mai important director al aplicaţiei şi conţine, la rândul lui, alte subdirectoare: `curbe Bezier`, `curbe ec param`, `suprafete Bezier`, `suprafete ec param` şi `library`. Fiecare dintre aceste subdirectoare, cu excepţia lui `library`, conţine toate fişierele necesare construirii tipului de obiect 3D precizat de numele subdirectoarelor.

Subdirectorul `library` conţine o bibliotecă de obiecte virtuale organizată astfel: `graphs` conţine toate fişierele *.wrl* (VRML) care reprezintă obiectele 3D, iar în `images` se află imagini în format *.gif* ale obiectelor din bibliotecă. Fişierul `library.xml` conţine informaţiile necesare despre fiecare obiect în parte, iar programul `index.php` preia construcţiile XML din `library.xml` şi le transformă într-o pagină HTML cu legături spre obiectele din bibliotecă.

3.2. Scrierea codului

Proiectul *GraphX3D* înglobează:

- fişiere *.php*, care asigură construirea dinamică a întregului sit;
- fişiere *.wrl*, constituind reprezentările 3D ale graficelor obiectelor din geometria în spaţiu. Ele sunt create dinamic, pornind de la funcţiile date de utilizator.
- fişiere *.svg*, care creează grafica sitului;
- fişiere *.xml*, în care se memorează informaţiile din sit. Ele sunt prelucrate de către funcţii PHP, pentru a putea fi afişate. Separarea dintre informaţii şi modul în care acestea sunt afişate face uşoară modificarea interfeţei şi a datelor;
- un fişier *.css*, care impune un stil uniform în tot situl;
- o bază de date MySQL, numită *graphX3D*. Aceasta conţine tabela *users*, unde se memorează utilizatorii înregistraţi şi preferinţele acestora.

În continuare vom menţiona limbajele utilizate în aplicaţie şi vom prezenta, pentru fiecare în parte, câteva aspecte interesante ale codului scris.

SVG

Întreaga grafică a sitului este realizată în *SVG* (*Scalable Vector Graphics*). După cum am precizat în prima secţiune, *SVG* este un format text bazat pe marcatori şi poate fi utilizat ca orice alt format XML.

Unul din avantajele SVG-ului este posibilitatea de a utiliza **scripturile**. SVG oferă suport pentru o gamă mare de evenimente ce pot fi utilizate alături de orice obiect grafic. Mai mult, scripturile pot acţiona asupra elementelor SVG şi XHTML simultan în cadrul unei aceleiaşi pagini Web.

Un exemplu în acest sens este programul JavaScript care actualizează în acelaşi timp culoarea unui element SVG şi textul unui câmp de tip *input* din fişierul HTML părinte. Scriptul este:

```
<script type="text/javascript"> <![CDATA[
   function circle_click(evt) {
       var circle = evt.target;
       var currentColour = circle.getAttribute('fill');
       var circle_final=document.getElementById('final');
       circle_final.setAttribute('fill', currentColour);
       var doc = self.parent.document;
              doc.form.colour.value=currentColour;
   }
]]> </script>
```

iar funcţia scrisă mai sus este apelată din fişierul SVG atunci când utilizatorul face *click* pe un cerc:

```
<circle onclick="circle_click(evt)" fill="#990033"/>
```

Codul funcţiei `circle_click(evt)` realizează următoarele: preia elementul (*target*) care a declanşat evenimentul *evt* şi ia valoarea atributului său 'fill' (memorează culoarea). Găseşte apoi în documentul curent (adică fişierul SVG) elementul cu *id*-ul 'final' şi îi setează culoarea ca fiind cea tocmai memorată. Aceasta înseamnă că, la *click*-ul utilizatorului, cercul care reprezintă culoarea finală îşi va schimba culoarea în cea selectată.

Funcţia caută apoi documentul părintelui documentului curent (`self.parent.document`) şi setează valoarea *input*-ului cu *id*-ul 'colour' la valoarea în hexazecimal a culorii curente.

Un alt script pe care l-am considerat interesant este cel care creează animaţia:

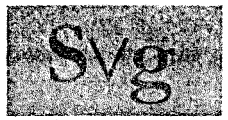
```
<script type="text/javascript"> <![CDATA[
   function text_over_out(idtext) {
          var svgobj=document.getElementById(idtext);
          var svgclass=svgobj.getAttribute("class");
          if(svgclass == "first")
             svgobj.setAttribute("class","second")
```

```
            if(svgclass == "second")
               svgobj.setAttribute("class","first")
      }
      function mouse_over(idtext) {
            var svgobj=document.getElementById(idtext);
            var svgstyle = svgobj.getStyle();
            svgstyle.setProperty ('visibility', 'visible');
      }
      function mouse_out(idtext) {
            var svgobj=document.getElementById(idtext);
            var svgstyle = svgobj.getStyle();
            svgstyle.setProperty ('visibility', 'hidden');
      } ]]> </script>
```

Funcţia `text_over_out` schimbă culoarea textului atunci când cursorul *mouse*-ului trece peste el. Realizează acest lucru prin preluarea obiectului cu *id*-ul 'idtext' şi setarea atributului 'class' al acestuia. Cum 'class' are precizată culoarea în fişierul CSS, textul îşi modifică înfăţişarea.

Cele două funcţii, `mouse_over` şi `mouse_out`, fac textul să apară, respectiv să dispară. Acest lucru se realizează foarte simplu, prin setarea atributului 'visibility' în 'hidden' sau 'visible'.

Specificaţia SVG cuprinde şi o serie de elemente declarative (text) pentru efecte de filtrare a obiectelor create. **Filtrele** pot schimba complet aspectul graficii, creând efecte de nori, marmură, ceaţă, surse de lumină, *blend* (combinarea a două obiecte), efecte 3D şi altele. Pentru a ilustra puterea de transformare a acestor filtre includem două imagini ale aceluiaşi fişier SVG, înainte şi după aplicarea filtrelor:



Înainte de aplicarea filtrului

După aplicarea filtrului

Exemplu de transformare a unui fişier SVG prin aplicarea unui filtru

Aplicaţia *GraphX3D* foloseşte un filtru pentru crearea de efect 3D, al cărui cod îl dăm mai jos:

```
<filter id="MyFilter">
1.<feGaussianBlur in="SourceAlpha" stdDeviation="4"
         result="blur"/>
2.<feOffset in="blur" dx="4" dy="4" result="offsetBlur"/>
3.<feSpecularLighting in="blur" surfaceScale="5"
         specularConstant=".75" specularExponent="20"
         lighting-color="#bbbbbb" result="specOut">
      <fePointLight x="-5000" y="-10000" z="20000"/>
```



```
  </feSpecularLighting>
4.<feComposite in="SourceGraphic" in2="specOut"
         operator="arithmetic" k1="0" k2="1" k3="1" k4="0"
         result="litPaint"/>
5.<feMerge>
      <feMergeNode in="offsetBlur"/>
      <feMergeNode in="litPaint"/>
  </feMerge>
</filter>
```

Prima primitivă a filtrului, *feGaussianBlur*, numerotată cu 1, acceptă ca intrare SourceAlpha, adică elementele de grafică reprezentate doar cu negru, şi stochează rezultatul într-un *buffer* numit 'blur'.

A două primitivă, *feOffset*, preia *buffer*-ul 'blur', mută rezultatul cu 4 pixeli pe axa *x* şi pe axa *y* şi stochează rezultatul într-un nou *buffer*, 'offsetBlur'. Se creează astfel un efect de umbră.

A treia primitivă, *feSpecularLighting*, utilizează *buffer*-ul 'blur' ca model pentru efectul 3D şi generează un efect de iluminare dintr-o singură sursă. Rezultatul este stocat în *buffer*-ul 'specOut'.

A patra primitivă, *feComposite*, compune rezultatul de la primitiva a treia cu obiectul grafic iniţial, obţinând un obiect colorat iluminat.

Ultima primitivă, *feMerge*, suprapune două straturi. Stratul din fundal constă din umbra rezultată de la primitiva a doua, iar stratul din faţă constă din obiectul iluminat de la primitiva 4.

Filtrul este aplicat unui grup de obiecte SVG prin atributul 'filter':

```
<g  filter="url(#MyFilter)">
      ...
</g>
```

VRML

Limbajul VRML este utilizat în aplicaţia de faţă pentru redarea reprezentării grafice a obiectelor 3D. Toate fişierele VRML sunt generate dinamic de funcţii PHP care au ca date de intrare ecuaţiile parametrice sau punctele de control ale obiectelor.

În general, pentru redarea curbelor s-a folosit nodul *Extrusion*, care ia o formă 2D (exprimată doar prin coordonatele *x* şi *y*) şi o translează de-a lungul unui drum (exprimat prin puncte 3D). Un exemplu de utilizare a acestui nod este:

```
geometry DEF Curba Extrusion {
   spine [ 0.53657291800043 0.84385395873249 -12,
           0.57807746141705 0.81598189232459 -11.95,
           ... ]
   crossSection [ -4 0, -3.873 1, ... ]
   scale [0.01 0.01]
}
```

Câmpul `crossSection` conţine descrierea formei ce trebuie translată, în cazul nostru un cerc. Lista `spine` este o listă de puncte 3D care formează linia de-a lungul căreia 'crossSection' este translată. `scale` este o listă de perechi de numere *float* (un număr pentru axa *x* şi unul pentru axa *y*) descriind scara la care este reprezentată 'crossSection' în fiecare punct. În exemplul de mai sus, în toate punctele se păstrează aceeaşi scară, de 0.01, în ambele direcţii.

Pentru redarea suprafeţelor s-au folosit nodurile *IndexedFaceSet* şi *IndexedLineSet*. Primul nod redă suprafaţa, iar al doilea descrie cadrul (*wireframe*) pe care structura este construită. Ca sintaxă, cele două noduri sunt destul de asemănătoare. Ambele au un câmp `coord`, care precizează punctele de construcţie, şi un câmp `coordIndex`, care descrie modul de unire a coordonatelor. Deosebirea este că primul nod desenează suprafeţe plane, pe când al doilea desenează linii. Un exemplu din codul generat de *GraphX3D* este:

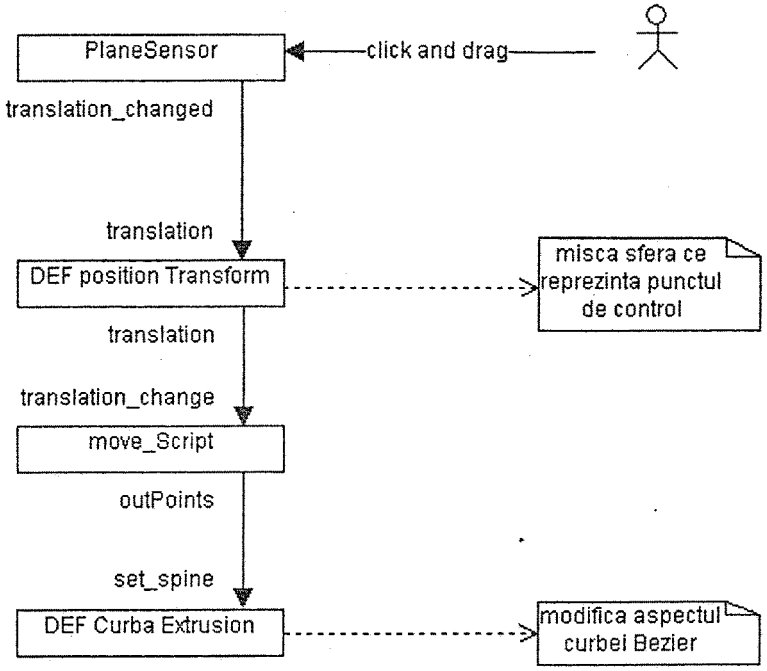
```
geometry IndexedFaceSet {
        coord Coordinate {
            point [ -0.0003185305 -0.1999997463 0.2, ...] }
        coordIndex [ 0 1 38 37 -1 1 2 39 38 -1 ...]
}
geometry IndexedLineSet {
        coord Coordinate {
            point [ -0.0003185305 -0.1999997463 0.2, ...] }
        coordIndex [0 1 -1 0 37 -1 1 38 -1 1 2 -1 2 39 -1 ...]
}
```

Numerele din câmpul `coordIndex` desemnează poziţia punctului în tabloul `point`. Cum nu există poziţii negative (primul element este la poziţia 0), '-1' desemnează sfârşitul unei linii/suprafeţe plane.

VRML oferă posibilitatea interacţiunii utilizatorului cu lumea 3D creată şi are suport pentru includerea de scripturi. Am folosit aceste două caracteristici pentru a face aplicaţia mai interesantă şi mai atrăgătoare pentru utilizator. Astfel, după redarea curbei Bézier, utilizatorul are posibilitatea de a modifica poziţia punctelor de control şi de a vedea cum influenţează acestea forma curbei. Tehnic, această interacţiune funcţionează astfel:

- pentru fiecare punct de control, reprezentat printr-o sferă, s-a definit un senzor de tip *PlaneSensor*, care poate fi mişcat de către utilizator prin lumea virtuală;
- evenimentul generat de *PlaneSensor* la schimbarea poziţiei, '*translation_changed*', este trimis ca *eventIn* punctului de control asociat, care îşi setează astfel valoarea câmpului *translation*. Efectul este că la schimbarea poziţiei unui *PlaneSensor*, se schimbă şi poziţia punctului de control;
- modificarea poziţiei punctului de control activează funcţia '*translation_change*' din scriptul scris în JavaScript. Aceasta recalculează punctele de pe curbă şi le stochează în '*outPoints*'.
- actualizarea câmpului '*outPoints*' determină setarea câmpului '*spine*' din nodul *Extrusion* care reprezintă curba la noile valori din '*outPoints*'.

Pentru a clarifica, schiţăm mecanismul de funcţionare:



Schema de funcţionare a animaţiei ce modifică forma curbei Bézier

Codul VRML care implementează această schemă, pentru un singur punct de control, este:

```
Shape {
   appearance Appearance { material Material { ... } }
   geometry DEF Curba Extrusion {
       spine [ ... ] crossSection [ ... ] scale [ ... ] }
}
DEF position Transform {
   translation -2 2 2
   children [
      Shape {
         appearance Appearance { material Material { ... } }
         geometry Sphere { radius .2 }
      }
      DEF PlaneS PlaneSensor { }
   ]
} #End Transform
```

```
DEF move_Script Script {
   eventIn  SFVec3f translation_change
   eventOut MFVec3f outPoints
   field SFVec3f P0 -2 2 2
   field SFFloat step 0.01
   field MFVec3f points [ . . .]
   url "javascript:
      function translation_change(value) {
         P0 = value;
         get_point(P0[0], P0[1], P0[2], P1[0], P1[1], P1[2],
             P2[0], P2[1], P2[2],P3[0], P3[1], P3[2]);
      outPoints = points;
   }
   function get_point(x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3) {
      k=0;
      for(u=0;u<=1;) {
         points[k][0] = x0*(1-u)*(1-u)*(1-u) +
                        3*x1*u*(1-u)*(1-u) +
                        3*x2*u*u*(1-u) + x3*u*u*u;
         points[k][1] = ... //la fel ca mai sus, dar pentru
                              y, nu x
         points[k][2] = ... //la fel ca mai sus, dar pentru z
         u = u+step; k++;
      } //end for
   } //end function
" } #end JavaScript si nodul Script
ROUTE PlaneS.translation_changed TO position.set_translation
ROUTE position.translation TO move_Script.translation_change
ROUTE move_Script.outPoints TO Curba.set_spine
```

Funcţia `get_point` calculează punctele de pe curbă utilizând polinoamele Bernstein.

Cuvântul-cheie ROUTE este folosit pentru a iniţia şi controla animaţia. El defineşte interconectarea dintre *eventOut*-ul nodului sursă şi *eventIn*-ul nodului destinaţie. Se observă că producerea unui *eventIn* al nodului Script determină accesarea funcţiei cu acelaşi nume ca *eventIn*-ul.

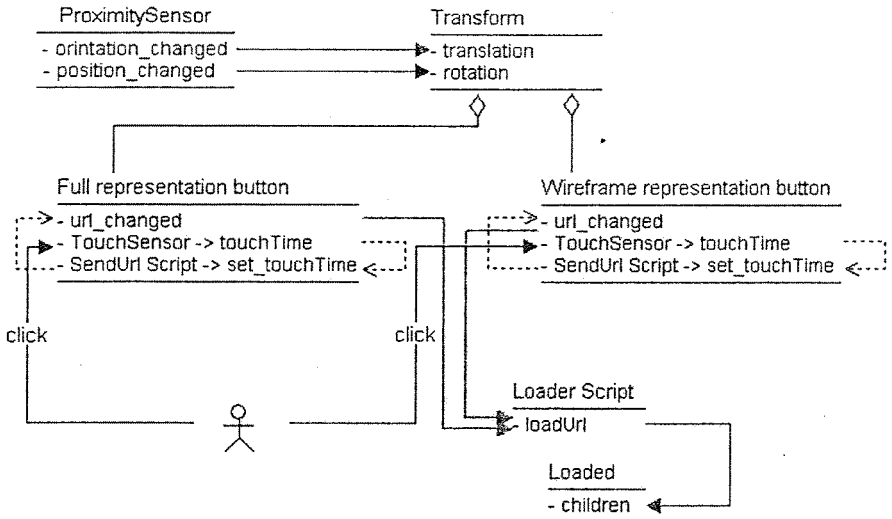
O altă implementare pe care o considerăm interesantă este cea a reprezentării grafice a suprafeţelor exprimate prin ecuaţii parametrice. Aici, utilizatorul poate vedea fie reprezentarea 'full' a suprafeţei (cu textura şi culoarea aplicate), fie reprezentarea 'wireframe' (cadrul pe care este construită suprafaţa şi punctele care o alcătuiesc).

Selectarea unei reprezentări se face printr-un *click* pe unul dintre cele două butoane. Butoanele, deşi se află în lumea virtuală, nu îşi schimbă poziţia o dată cu celelate obiecte. Acest fapt este determinat de folosirea senzorului *ProximitySensor*. Acest nod defineşte o zonă de senzori sensibilă la mişcările utilizatorului în spaţiul virtual.

În momentul când utilizatorul se roteşte sau îşi schimbă poziţia, *ProximitySensor* generează evenimente de tip *eventOut* care sunt transmise nodului *Transform* care conţine butoanele. În urma primirii ca *eventIn* a noii poziţii şi orientări a utilizatorului, nodul *Transform* îşi setează câmpurile astfel încât să apară nemişcat pentru privitor.

Selectarea uneia dintre cele două reprezentări se face prin încărcarea/ştergerea de obiecte din lumea virtuală. Practic, există două fişiere, *full.wrl* şi *wireframe.wrl*, din care fişierul de bază, *result.wrl*, îşi încarcă informaţia. Fiecare buton are asociat un *URL* (primul buton reţine fişierul *full.wrl*, al doilea fişierul *wireframe.wrl*) şi un *TouchSensor*, care produce evenimentul *touchTime* în momentul când utilizatorul dă *click* pe buton. Evenimentul *touchTime* determină trimiterea URL-ului la funcţia *loadUrl()* din nodul de tip Script. Această funcţie încarcă fişierul primit şi îl adaugă la nodurile din fişierul curent.

Schema de rutare a mecanismului descris mai sus este:



Schema de rutare a mecanismului de selectare a reprezentărilor *full* şi *wire-frame* ale suprafeţei

XML

Limbajul XML este utilizat în aplicaţia *GraphX3D* pentru a stoca informaţiile într-un mod structurat, care să permită prelucrarea simplă şi flexibilă a datelor. Prin separarea realizată astfel între textul informaţional şi modalitatea de prezentare a lui, viitoarele modificări de design sau de informaţie vor fi mult mai uşor de realizat.

În general, structura documentelor XML din aplicaţia *GraphX3D* este următoarea:

```
<doc>
   <title>Titlul paginii</title>
```

```
    <chapter>
        <subtitle>Titlul capitolului / sectiunii</subtitle>
        <idea>paragraf simplu</idea>
        <idea> ... </idea> ...
    </chapter>
    <chapter> ... </chapter> ...
</doc>
```

Această structură este potrivită pentru informaţiile teoretice, prezentate într-un format asemănător celui din cărţi, dar pentru biblioteca de obiecte 3D el nu este ideal. De aceea, formatul fişierului `library.xml` este altul:

```
<library>
    <function>
        <name> numele functiei </name>
        <graph file="numele fisierului wrl">
                <image> numele imaginii functiei </image>
        </graph>
        <description> descriere a functiei </description>
    </function>
    <function>
        ...
    </function>
    ...
</library>
```

Pe baza acestei structuri, fişierul *library.php* construieşte un tabel în HTML, care are drept coloane numele funcţiei, o imagine a graficului acesteia şi o scurtă descriere a funcţiei. Numele şi imaginea constituie o legătură la fişierul *.wrl* care conţine reprezentarea grafică a funcţiei. Detalii despre cum se construieşte din XML fişierul HTML sunt date în secţiunea dedicată limbajului PHP.

PHP

Limbajul PHP este utilizat pentru a asigura construirea dinamică a întregului sit. Am ales PHP pentru că a fost realizat special pentru dezvoltarea aplicaţiilor Web, poate fi inclus în fişierele HTML şi este disponibil gratuit pentru platforme ca UNIX/Linux şi Windows.

După cum am precizat mai sus, fişierele PHP din *GraphX3D* generează codul VRML necesar reprezentării obiectelor 3D. De asemenea, tot prin PHP se realizează conectarea la baza de date şi procesarea fişierelor XML.

Prelucrarea documentelor XML se realizează cu ajutorul unui analizor XML pus la dispoziţie de limbajul PHP. Analiza XML este bazată pe evenimente ce trebuie tratate de funcţii definite de programator, fiecare construcţie din documentul XML generând un eveniment. De asemenea, PHP are o serie de funcţii predefinite ce se pot utiliza în definirea funcţiilor pentru tratarea evenimentelor, cum ar fi `xml_parser_create()` – pentru iniţializarea analizorului XML –, `xml_set_element_handler()` – pentru *tag*-urile de început şi sfârşit – şi altele. O descriere completă a acestor funcţii se găseşte în cartea *Tehnologii Web* scrisă de Sabin Buraga (a se vedea partea G din volumul II). Clasa `parseXML`, folosită şi de aplicaţia *GraphX3D*, pentru la procesarea documentelor XML este disponibilă în anexa C a cărţii de faţă.

O soluţie alternativă este cea prezentată în capitolul 4, prin utilizarea bibliotecii XML22 (a se consulta şi anexa D).

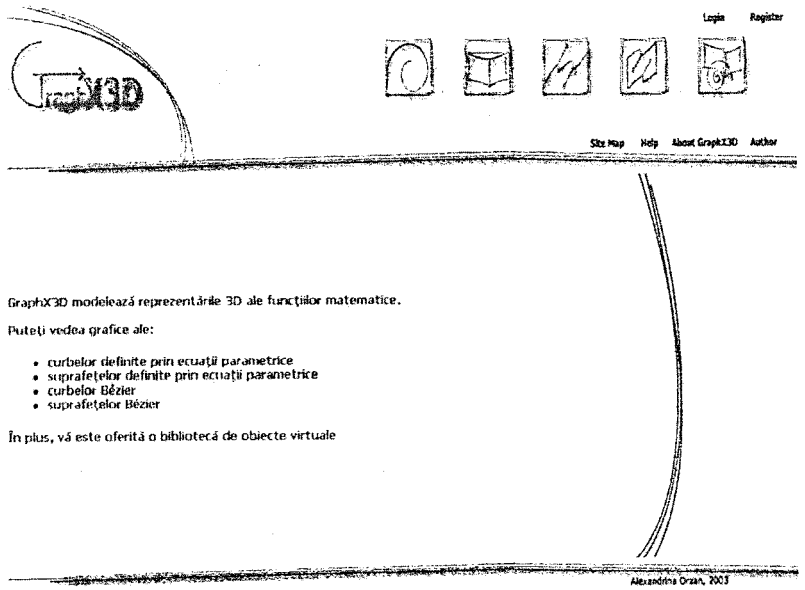
Conectarea la baza de date şi manipularea informaţiilor din tabelul *users* se realizează prin funcţii de acces al serverelor de MySQL specificate în limbajul PHP. Acestea sunt `mysql_pconnect()` – realizează o conexiune persistentă la server-ul MySQL –, `mysql_select_db()` – selectează baza de date – şi `mysql_query()` – interoghează baza de date.

4. Manual de utilizare

În acestă secţiune vom prezenta aplicaţia *GraphX3D* din perspectiva utilizatorului. Prima subsecţiune conţine descrierea succintă a unor părţi importante din aplicaţie, iar a doua subsecţiune precizează cerinţele hardware şi software necesare pentru aplicaţie.

Pagina principală a aplicaţiei conţine o scurtă prezentare a ceea ce face situl şi legături către paginile importante din sit. Utilizatorul poate să aleagă tipurile de obiecte 3D pentru care vrea să vadă reprezentarea grafică sau să viziteze biblioteca de obiecte virtuale.

O captură-ecran cu pagina principală a aplicaţiei se poate vedea în figura de mai jos.



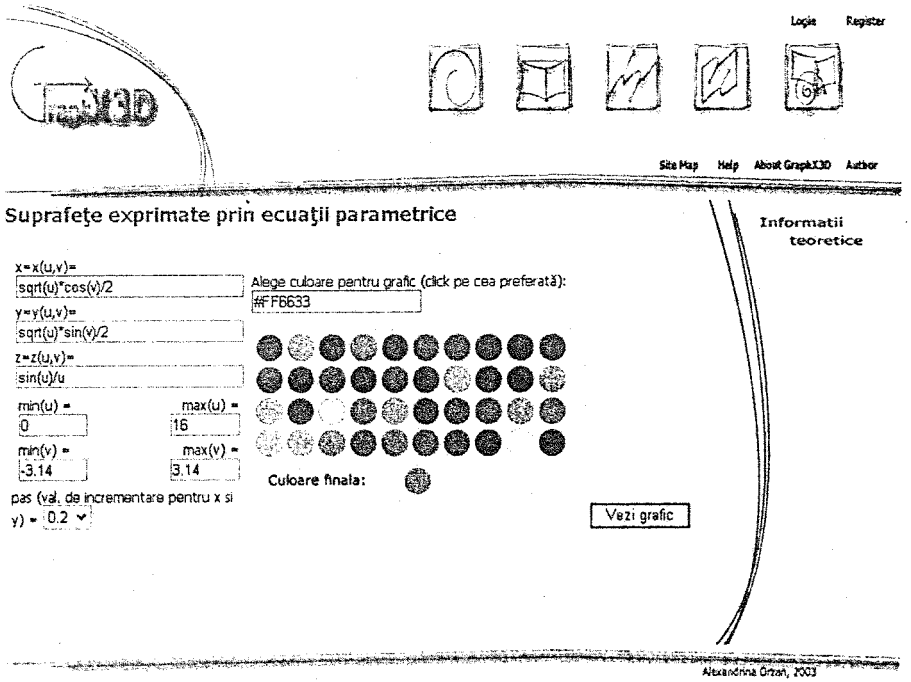
Pagina de început a sitului *GraphX3D*

Atunci când utilizatorul alege unul din tipurile de obiecte geometrice oferite, se încarcă o pagină în care vizitatorul este invitat să completeze ecuaţiile parametrice sau puntele de control ale obiectului ales. El trebuie să aleagă şi culoarea care va fi folosită la realizarea graficului. De asemenea, aici se găseşte şi o legătură către o prezentare teoretică a clasei de obiecte alese.

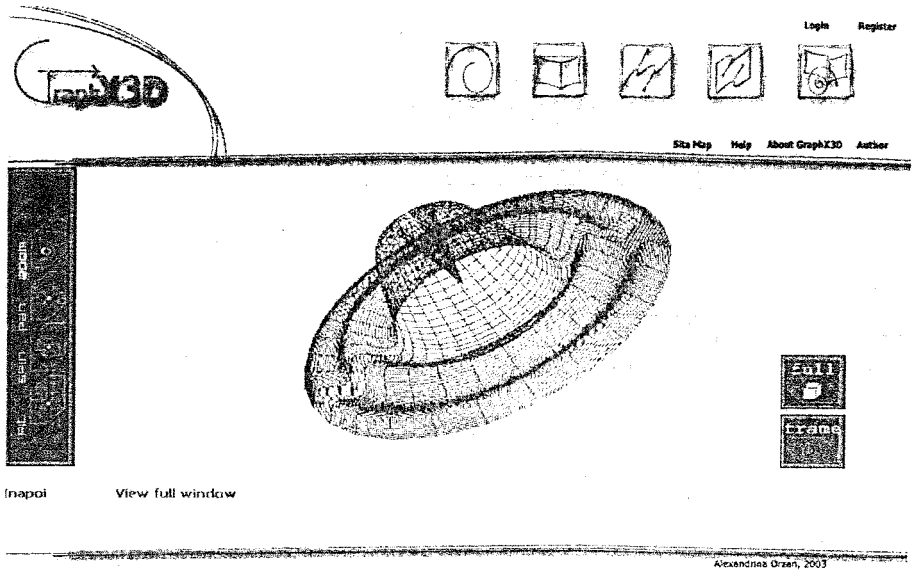
După ce toate datele au fost introduse, utilizatorul trebuie să dea *click* pe butonul **Vezi grafic**. Aplicaţia noastră verifică dacă toate informaţiile sunt corecte. În cazul unor date introduse eronat, se afişează un mesaj de eroare, precizându-se care anume sunt greşite. Dacă toate datele sunt corecte, aplicaţia afişează graficul. Pentru o manipulare mai bună a graficului 3D, utilizatorul poate să aleagă să vadă obiectul virtual într-o fereastră nouă apăsând **View full window**.

La reprezentarea grafică a suprafeţelor definite prin ecuaţii parametrice, utilizatorul poate să vizualizeze suprafaţa obţinută fie cu textura aplicată, fie în formă de „cadru de sârmă" (*wireframe*). Curbele şi suprafeţele Bézier pot fi modificate, în sensul că punctele de control pot fi mutate cu *mouse*-ul, iar forma curbei/suprafeţei se va schimba corespunzător.

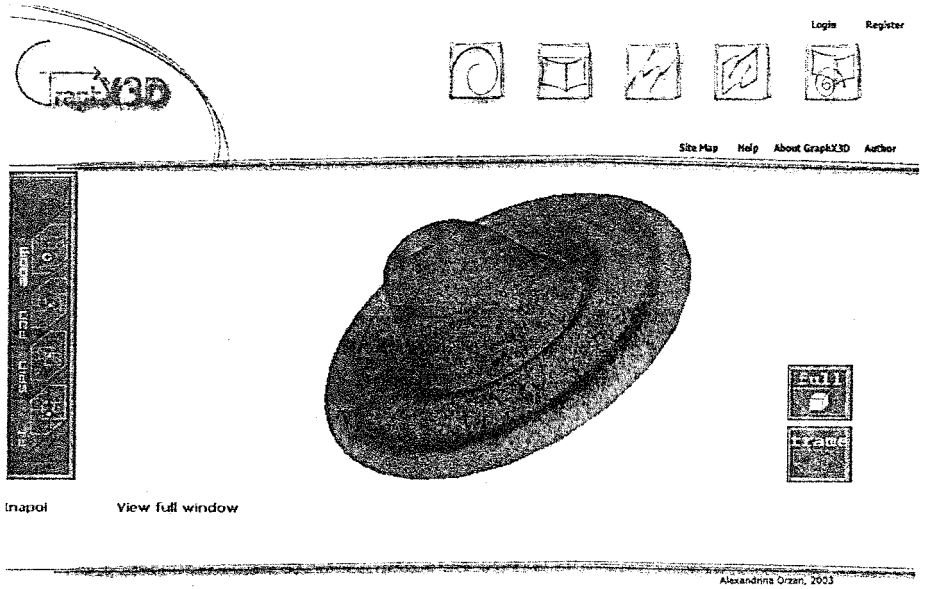
Următoarele capturi-ecran ilustrează un exemplu de interfaţă pentru introducerea de date şi modul de reprezentare a obiectelor 3D.



Pagina pentru introducerea ecuaţiilor parametrice a suprafeţelor



Reprezentarea *wire-frame* a suprafeţei exprimate prin ecuaţiile parametrice definite mai sus

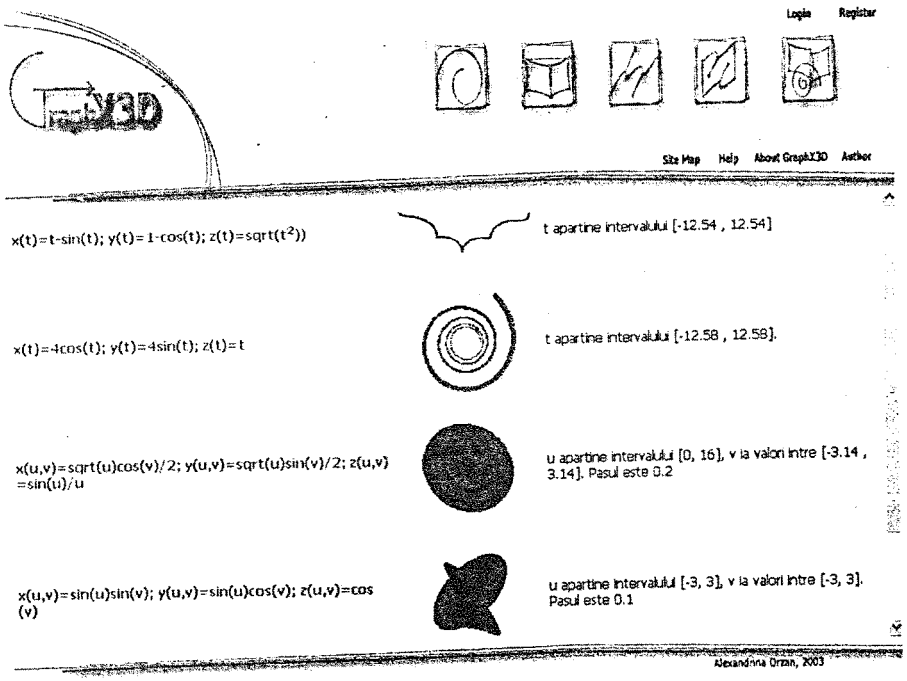


Reprezentarea completă (*full*) a suprafeţei

Ecuaţiile introduse în imaginea de mai sus sunt:

```
x = x(u,v) = sqrt(u)*cos(v)/2;
y = y(u,v) = sqrt(u)*sin(v)/2;
z = z(u,v) = sin(u)/u,                    cu 0<u<16 si -3.14<v<3.14
```

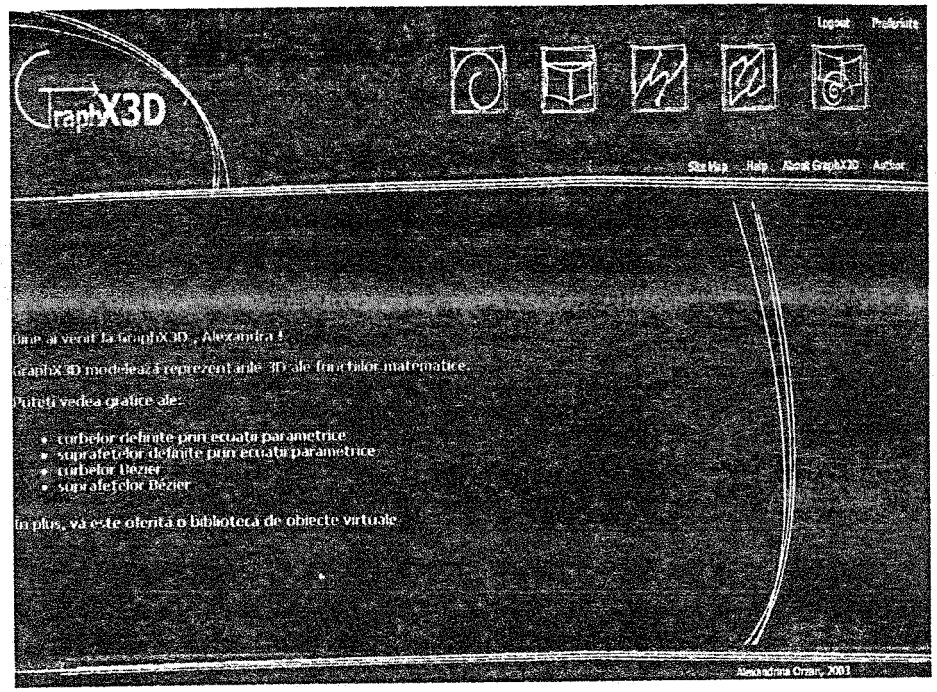
Dacă utilizatorul alege să viziteze biblioteca virtuală, atunci se deschide o pagină care prezintă, pentru fiecare obiect în parte, funcţiile din care a fost obţinut, o imagine a graficului şi unele informaţii suplimentare despre obiectul respectiv. Dacă utilizatorul dă *click* pe funcţie sau pe imagine, atunci o nouă fereastră, cu reprezentarea *.wrl* (VRML) a obiectului, se deschide. O parte a bibliotecii virtuale se poate vedea în figura următoare:



Biblioteca de obiecte virtuale

În afară de informaţiile matematice prezente în sit, utilizatorul poate să parcurgă harta sitului, să citească secţiunea de *help* şi să afle informaţii despre sit şi autorul lui. Mai mult, el poate să contacteze autorul pentru orice reclamaţie sau nelămurire.

Utilizatorului îi este oferită, în cadrul situl nostru, posibilitatea de a-şi crea un cont. Vizitatorul poate astfel să-şi precizeze preferinţele pentru design-ul sitului. Poate să modifice culoarea de fundal, culorile graficii şi culoarea textului. Aspectul sitului *GraphX3D* pentru un utilizator care şi-a definit preferinţele se poate vedea în figura următoare:



Aspectul sit-ului *GraphX3D* pentru un utilizator care şi-a definit noi preferinţe

Cerinţe software

Pentru o bună exploatare a aplicaţiei, utilizatorul trebuie să aibă instalate două vizualizatoare (disponibile gratuit), unul pentru SVG şi unul pentru VRML. Pentru SVG recomandăm vizualizatorul produs de Adobe, care se poate găsi la adresa **http://www.adobe.com/svg/viewer/install/main.html**, iar pentru VRML, recomandăm vizualizatorul produs de Parallel Graphics: **http://www.parallelgraphics.com.**

Sistemul care găzduieşte aplicaţia *GraphX3D* trebuie să aibă instalate serverul PHP cel puţin versiunea 4.1.1 şi serverul de baze de date MySQL (utilizat pentru înregistrarea utilizatorilor şi a preferinţelor acestora în baza de date *GraphX3D*).

Cerinţe hardware

Aplicaţia nu are nevoie de dispozitive hardware speciale.

Situl a fost accesat în perioada de testare de pe un calculator cu procesor AMD Duron 1GHz, 256 MB de DDRAM, placă video NVIDIA GeForce2 MX şi monitor de 17" şi de pe un calculator cu procesor Pentium 200MHz, cu 32M de SDRAM şi monitor de 15". Nu s-au sesizat diferenţe majore în redare.

Comentarii bibliografice

Unele detalii privind tehnologiile utilizate în cadrul aplicaţiei *GraphX3D* (ca de exemplu, crearea de lumi VRML sau procesarea fişierelor XML în PHP) se pot consulta în *Tehnologii Web*, de Sabin Buraga. Despre manipularea fişierelor XML a se vedea şi articolul „PHP, programare obiectuală şi XML" apărut în *NET Report*, martie 2001 (disponibil *on-line* la **http://www.infoiasi.ro/~busaco/publications/articles/php-oop-xml.pdf**).

Cititorul interesat de realitatea virtuală şi de VRML poate vizita situl Consorţiului Web3D, **www.web3d.org**, care oferă informaţii actuale despre lumea Web-ului tridimensional şi multe legături utile către tutoriale şi *browsere* pentru VRML şi X3D. O altă colecţie impresionantă de resurse, incluzând tutoriale, aplicaţii şi soft pentru dezvoltarea lumilor 3D, se află la adresa **www.lighthouse3d.com**. O referinţă foarte bună o constituie cartea A. Ames, D. Nadeau, J. Moreland, *VRML 2.0 Sourcebook*, John Wiley & Sons, 1997.

Pentru un exemplu de utilizare a limbajului VRML puteţi vizita situl **www.outline3d.com**, care oferă posibilitatea de a proiecta mobilă şi de a realiza design-uri interioare în manieră *on-line*.

Unelte pentru dezvoltarea de lumi VRML găsiţi (pentru platforma Windows), la adresele **www.vizx3d.com** şi **www.parallelgraphics.com**.

Pentru informaţii referitoare la SVG, adresa principală este **www.w3.org/Graphics/SVG**. Aici se află specificaţia completă, unelte de dezvoltare, vizualizatoare, articole despre SVG şi informaţii de ultimă oră. A se consulta şi Vincent Hardy, „Scalable Vector Graphics (SVG) – An Executive Summary", *Sun Articles*, 2003: **http://wwws.sun.com/software/xml/developers/svg/** şi Eric Vitiello Jr., „SVG: The New Flash", *Digital Web Magazine*, 2003: **http://www.digital-web.com/index.shtml**.

Dacă doriţi să experimentaţi funcţionarea aplicaţiei GraphX3D, puteţi vizita adresa **http://students.infoiasi.ro/~alexo/graphX3D**.

CAPITOLUL 6

Dynamic Charts – grafice în PHP

Cristina Oprea, cristinaois@yahoo.com
Sabin Buraga, busaco@infoiasi.ro

Capitolul prezintă o aplicaţie Web de generare dinamică a unor grafice (*charts*) asemănătoare celor realizate de programele de calcul tabelar, folosind funcţiile de prelucrare grafică puse la dispoziţie de biblioteca GD. Accesul la funcţionalităţile bibliotecii se poate face direct din limbajul PHP prin intermediul unor funcţii predefinite, utilizarea acestora fiind deosebit de comodă pentru programator. Implementarea aplicaţiei respectă paradigma programării orientate-obiect, bazându-se pe suportul obiectual oferit de PHP.

1. Preambul

Capitolul de faţă prezintă aplicaţia *Dynamic Charts* – o aplicaţie Web care creează imagini grafice în mod dinamic, oferind posibilitatea alegerii modelelor de grafice (*charts*) dintr-o listă predefinită. Aceste tipuri de grafice pot fi folosite, de exemplu, pentru analiza statistică a datelor, utilizarea lor realizându-se prin intermediul unui sit Web care va pune la dispoziţie şi facilităţi de salvare a datelor utilizate, pentru o refolosire ulterioară.

Aplicaţia a fost realizată în PHP, codul-sursă fiind orientat-obiect şi recurgând la funcţiile grafice ale bibliotecii *GD* (versiunile 1.8.4 şi 2.0.1) pentru generarea fişierelor PNG corespunzătoare graficelor dorite de utilizator. Astfel, *Dynamic Charts* poate fi considerat un exemplu concludent de utilizare a facilităţilor de creare şi de manipulare avansată a unui set de primitive grafice menite a realiza dinamic imagini direct pe Web. Aplicaţia de faţă este o alternativă la biblioteca *JpGraph* utilizată de situl *E-Pizza* (a se vedea capitolul 4 şi anexa E).

După o prezentare a bibliotecii GD şi a principalelor caracteristici ale funcţiilor pe care le pune la dispoziţia programatorului, capitolul va continua cu descrierea generală a aplicaţiei *Dynamic Charts*, oferindu-se diverse detalii privitoare la unele aspecte de implementare şi de utilizare a codului PHP.

2. Biblioteca GD

2.1. Caracterizare şi scurt istoric

Serverul de aplicaţii PHP oferă numeroase facilităţi de prelucrare a imaginilor, punând la dispoziţie o pleiadă de funcţii pentru desenarea de puncte, linii, poligoane, arce de cerc, pentru controlul culorii şi fonturilor, oferind în plus posibilitatea de a încărca şi salva imagini în popularele formate *GIF* (*Graphics Interchange Format*), *PNG* (*Portable Network Graphics*) sau *JPEG* (*Joint Picture Experts Group*). Singura cerinţă este să fie instalată biblioteca *GD* în sistem.

GD reprezintă o bibliotecă de funcţii grafice care permite:

- desenarea rapidă de imagini care conţin linii, arce, texte folosind diverse culori;
- tăierea şi lipirea unor părţi din alte imagini;
- umplerea unor porţiuni;
- afişarea rezultatului ca fişiere GIF, PNG sau JPEG (formatele grafice cele mai populare la ora actuală în spaţiul WWW).

GD nu este un program de „pictură" (gen *Adobe Photoshop*, *Corel Draw!* sau *GIMP*), aşa cum ar putea crede unii cititori, ci este un set de funcţii, recomandat în special programatorilor care doresc să genereze dinamic diverse ilustraţii, scheme şi figuri în cadrul unor aplicaţii. GD necesită cunoştinţe adecvate de programare. Din păcate însă, nu oferă chiar toate posibilităţile dorite în operaţiunile grafice, dar pune la dispoziţie facilităţi cu ajutorul cărora se poate îmbunătăţi activitatea de manipulare a imaginilor şi de consultare a diferitelor proprietăţi ale fişierelor grafice.

Scurt istoric

Prima bibliotecă GD permitea utilizatorilor manipularea fişierelor GIF. Sursa acelei prime versiuni a fost dezvoltată de către Thomas Boutell de la Boutell.Com, Inc şi a fost disponibilă gratuit pe Internet. În 1996, Unisys, care deţine drepturi de autor pentru algoritmii de compresie LZW – folosiţi în compresia fără pierderi a fişierelor GIF –, s-a decis să-şi impună aceste drepturi. Rezultatul a fost că biblioteca GD, disponibilă gratuit în format sursă, proiect liber de dezvoltare, nu a mai putut folosi formatul GIF. Astfel, întregul suport pentru generarea de imagini GIF a fost eliminat începând cu versiunea 1.6 a bibliotecii, cea mai bună soluţie găsită fiind folosirea imaginilor în formate moderne cum ar fi PNG şi JPEG.

De la prima versiune până în prezent s-au adăugat numeroase funcţii care au îmbunătăţit maniera de prelucrare a imaginilor, versiunea 1.7.3 având deja încorporate cele mai multe dintre caracteristicile cerute de obicei pentru pachete de prelucrare 2D pe 8 biţi, iar versiunea 2.0.1 includând suport pentru palete de culori, precum şi pentru imagini *TrueColor*. La dezvoltarea bibliotecii GD au participat programatori din întreaga lume, fiecare având o contribuţie importantă, lucru recunoscut şi de Thomas Boutell, cel care întreţine actualmente biblioteca.

Domenii de utilizare

Există mai multe limbaje de programare care pot folosi fără probleme biblioteca grafică GD:

- Perl – datorită programatorului Lincoln Stein este disponibil un modul denumit *GD.pm*, care foloseşte biblioteca GD ca bază pentru o mulţime de clase ce pot fi utilizate direct în Perl;
- Tcl – poate folosi de asemenea GD, datorită încărcării dinamice a pachetului de extensii *Gdtclft*;
- Pascal – există un pachet elaborat de Michael Brandbury numit *gdfp*;
- Hasskell şi Rexx – sunt disponibile, de asemenea, interfeţe GD şi pentru aceste limbaje.

La momentul actual există cel puţin trei interpretoare care pot executa operaţiile oferite de biblioteca GD. Vom vedea ulterior că PHP oferă, prin intermediul unor funcţii predefinite, toate facilităţile bibliotecii.

Restricţii în utilizarea bibliotecii GD

Începând cu versiunea 1.6, datorită lucrului cu formatul PNG, este nevoie de o bibliotecă de compresie *zlib* şi de biblioteca *libpng*. De la versiunea 1.6.2 există posibilitatea desenării textelor folosind fonturile *TrueType* pentru care trebuie instalată biblioteca *libttf*. Pentru a putea folosi fonturi *PostScript Type 1* trebuie instalată biblioteca *T1lib*. Aceste biblioteci se folosesc pe o varietate de platforme şi se găsesc gratuit pe Internet (de cele mai multe ori ele sunt disponibile în orice distribuţie Linux actuală). De asemenea, trebuie să existe în sistem un program de vizualizare a fişierelor PNG sau un navigator cum ar fi Netscape 4.04 ori o versiune superioară sau Microsoft Internet Explorer 4.0 sau versiuni mai recente, aceasta pentru vizualizarea rezultatelor în urma folosirii bibliotecii GD.

Avizăm cititorul că în cazul în care foloseşte versiuni mai vechi de GD 1.6, nu va exista suport decât pentru formatul GIF.

Pentru PHP, există o interfaţă a bibliotecii GD care actualmente este introdusă chiar în distribuţia oficială. În cazul Windows, singura cerinţă este includerea în fişierul *PHP.ini* a bibliotecii *php_gd.dll* (în cadrul secţiunii *Windows Extensions*, trebuie inserată o linie `extension=php_gd.dll`).

În general toate interfeţele bibliotecii GD specifice unui limbaj de programare folosesc aceleaşi funcţii de bază, dar adaptate la restricţiile sintactice impuse de limbajul respectiv. Un exemplu de creare a unei imagini în Perl, PHP şi respectiv C este următorul:

```
$im = new GD::Image (74, 33);        # Perl
$im = imagecreate (74, 33);          // PHP
im  = gdImageCreate (74, 33);        /* C */
```

2.1. Funcţii PHP pentru utilizarea bibliotecii GD

În continuare vom prezenta cele mai utilizate funcţii pe care le pune la dispoziţie biblioteca GD pentru PHP. Unele funcţii oferite de PHP vor putea funcţiona fără a fi în mod necesar instalată biblioteca GD (de exemplu, cele de consultare a informaţiilor EXIF inserate de camerele digitale în fişierele JPEG – a se vedea numele de funcţii prefixate de şirul de caractere *exif* din tabelul de mai jos).

Funcţii de creare, distrugere, încărcare şi salvare a imaginilor

int imagecreate (int x_size, int y_size) va crea o imagine de dimensiuni precizate de programator în *x_size* şi *y_size* pixeli, şi va returna un identificator de imagine asemănător cu un identificator de fişiere (*handler*) returnat de primitivele *create()* sau *open()*. Toate celelalte funcţii de prelucrare a imaginii vor folosi acest identificator (situaţie similară celei de la funcţiile de manipulare a fişierelor).

```
<?php
// vom trimite navigatorului o imagine PNG
header ("Content-type: image/png");
$im = @ImageCreate (300, 100)
    or die ("Eroare de initializare GD");
$bc = ImageColorAllocate
                        ($im, 255, 255, 255); // fundal alb
// culoarea textului va fi o nuanta de rosu
$tc = ImageColorAllocate ($im, 255, 33, 74);
ImageString ($im, 2, 24, 20,
   "Un salut din PHP cu GD", $tc);
// se genereaza si un dreptunghi
ImageRectangle($im, 10, 10, 50, 50, $tc);
// se trimite efectiv imaginea
ImagePng ($im);
?>
```

Reamintim cititorului că dacă prefixăm cu simbolul „@" numele unei funcţii, atunci posibilele mesaje de eroare întoarse de acea funcţie nu apar la ieşirea standard şi deci nu vor fi afişate utilizatorului final. În cazul exemplului dat mai sus, vom semnala eroarea printr-un mesaj în limba română, întrerupând – prin apelul funcţiei *die()* – execuţia programului.

int imagecreatefromgif (string filename) va încărca o imagine dintr-un fişier în format GIF. Funcţia va returna un identificator de imagine în caz de succes sau 0 în caz de eroare (de cele mai multe ori din cauză că fişierul este defect sau nu conţine o imagine în format GIF).

int imagecreatefrompng (string filename) va crea o imagine pornind de la un fişier în format PNG (funcţia este similară celei de mai sus).


```
// functie de incarcare a unui fisier PNG
// cu raportarea eventualelor erori survenite
function LoadPNG ($imgname) {
  $im = @ImageCreateFromPNG ($imgname);
  /* deschidem fisierul */
  if (!$im) { /* verificam daca s-a reusit deschiderea */
     $im  = ImageCreate (150, 30);
     /* am creat o imagine cu fundal alb */
     $bc  = ImageColorAllocate ($im, 255, 255, 255);
     $tc  = ImageColorAllocate ($im, 0, 0, 0);
     ImageFilledRectangle ($im, 0, 0, 150, 30, $bc);
     /* scriem eventualele erori */
     ImageString ($im, 1, 5, 5,
        "Eroare de incarcare $imgname", $tc);
    }
    return $im;
}
```

int imagecreatefromjpeg (string filename) va încărca o imagine dintr-un fişier în format JPEG.

int imagecreatefromstring (string image) va returna un identificator de imagine creată dintr-un şir de caractere.

int imagecreatefromwbmp (string filename) va crea o imagine pornind de la un fişier în format WBMP folosit de navigatoarele WAP. Ca şi funcţiile precedente, va returna un identificator de imagine în caz de succes sau 0 în caz de eroare.

resource imagecreatetruecolor (int x_size, int y_size) este asemănătoare funcţiei *imagecreate()* cu deosebirea că va crea imagini *TrueColor*, cu un număr „nelimitat" de culori.

int imagedestroy (int im) distruge identificatorul de imagine şi eliberează zona de memorie alocată informaţiilor despre acesta.

int imagegif (int im [, string filename]) trimite imaginea desemnată de un identificator de imagine la ieşirea standard (se trimite direct navigatorului) sau o salvează într-un fişier stocat în sistemul de fişiere al serverului Web. Implicit formatul va fi GIF87a. Dacă s-a setat proprietatea de transparenţă prin *imagecolortransparent()*, imaginea va fi generată în formatul GIF89a. În prezent, această funcţie este considerată demodată.

int imagepng (int im [, string filename]) trimite imaginea desemnată de un identificator de imagine la ieşirea standard (se trimite direct navigatorului Web) sau o salvează într-un fişier PNG stocat în sistemul de fişiere al serverului Web.

int imagejpeg (int im [, string filename [, int quality]]) expediază imaginea desemnată de un identificator de imagine la ieşirea standard (se trimite direct navigatorului) sau o salvează într-un fişier PNG. Se dă posibilitatea programatorului de a stabili şi

coeficientul de compresie cu pierderi a imaginii rezultate prin parametrul *quality* care poate lua valori în intervalul 0-100 (0 – compresie maximă, 100 – compresie scăzută [deci generarea unei imagini de calitate]; implicit se consideră valoarea 100).

Funcţii de desenare

int imagesetpixel (int im, int x, int y, int col) va desena un pixel de o anumită culoare la coordonatele specificate. Punctul de coordonate (0,0) se consideră a fi punctul din stânga-sus. Culoarea va fi un întreg desemnând un identificator de culoare returnat de funcţia *imagecolorallocate()*.

int imageline (int im, int x1, int y1, int x2, int y2, int col) va reprezenta un segment de dreaptă dat de coordonatele a două puncte, cu o anumită culoare.

int imagedashedline (int im, int x1, int y1, int x2, int y2, int col) va reprezenta un segment de dreaptă întreruptă dat de coordonatele a două puncte, cu o anumită culoare.

int imagerectangle (int im, int x1, int y1, int x2, int y2, int col) va desena un dreptunghi de culoare *col* în imaginea *im* începând de la coordonatele punctului din stânga sus *(x1, y1)* şi la coordonatele punctului din dreapta jos *(x2, y2)*.

int imagepolygon (int im, array points, int num_points, int col) va desena un poligon, coordonatele punctelor fiind furnizate prin intermediul tabloului *points*, iar numărul punctelor este dat de *num_points*.

int imagearc (int im, int cx, int cy, int w, int h, int s, int e, int col) va desena o elipsă parţială de culoare *col* centrată în *cx, cy*; parametrii *w* şi *h* indică lăţimea şi respectiv înălţimea elipsei, iar *s* şi *e* sunt valori specificate în grade ale punctelor între care se reprezintă arcul.

int imageellipse (resource im, int cx, int cy, int w, int h, int col) va desena o elipsă de culoare *col* centrată în punctul *cx, cy*; argumentele *w* şi *h* indică lăţimea şi respectiv înălţimea elipsei.

int imagefill (int im, int x, int y, int col) reprezintă o funcţie de umplere cu o anumită culoare pornind de la coordonatele precizate.

int imagefilledarc (int im, int cx, int cy, int w, int h, int s, int e, int col, int style) desenează o elipsă parţială, dar în plus permite umplerea ei cu o culoare în funcţie de argumentul *style* care poate avea următoarele valori:

1. IMG_ARC_PIE – produce o margine rotunjită;
2. IMG_ARC_CHORD – leagă unghiul de start şi de sfârşit printr-o linie dreaptă;
3. IMG_ARC_NOFILL – indică faptul că arcul şi corzile nu trebuie colorate, ci doar reprezentate prin linii;
4. IMG_ARC_EDGED – folosit împreună cu IMG_ARC_NOFILL indică faptul că unghiul de start şi de sfârşit trebuie legat de centru, ceea ce dă posibilitatea umplerii.

Această funcţie este disponibilă începând cu PHP versiunea 4.0.6, folosind biblioteca GD 2.0.1 ori o versiune ulterioară.

int imagefilledellipse (resource im, int cx, int cy, int w, int h, int col) desenează o elipsă de centru şi dimensiuni date, pe care o umple cu o culoare dată.

int imagefilledpolygon (int im, array points, int num_points, int col) desenează şi umple un poligon.

int imagefilledrectangle (int im, int x1, int y1, int x2, int y2, int col) creează şi umple un dreptunghi.

int imagefilltoborder (int im, int x, int y, int border, int col) umple o porţiune delimitată de culoarea *border*, începând cu punctul de coordonate *x* şi *y*, cu culoarea *col*.

int imagechar (int im, int font, int x, int y, string c, int col) va desena un caracter la poziţia specificată; argumentul *font* poate lua valori de la 1 la 5 şi reprezintă mărimea fontului (corpului de literă) folosit.

int imagecharup (int im, int font, int x, int y, string c, int col) va desena vertical un caracter la poziţia specificată de *x* şi *y*.

int imagestring (int im, int font, int x, int y, string s, int col) desenează şirul *s* în punctul (*x*, *y*), folosindu-se culoarea *col*.

int imagestringup (int im, int font, int x, int y, string s, int col) inserează în imagine în poziţie verticală şirul *s*.

Funcţii de manipulare a culorilor

int imagecolorallocate (int im, int red, int green, int blue) va aloca un identificator de culoare pentru imagine, pornind de la componentele RGB corespunzătoare (trei întregi între 0 şi 255 reprezentând componentele roşu, verde şi albastru, similar convenţiilor de culoare folosite de foile de stiluri Cascading Style Sheet – CSS).

Prima culoare alocată va fi considerată drept culoare de fundal pentru imaginea respectivă, dacă nu se specifică în mod explicit una.

```
$alb = ImageColorAllocate ($im, 255, 255, 255);
$verde = ImageColorAllocate ($im, 0, 255, 0);
$gri = ImageColorAllocate ($im, 127, 127, 127);
```

int imagecolordeallocate (int im, int index) va elibera un identificator de culoare alocat în prealabil de funcţia anterioară.

int imagecolortransparent (int im [, int col]) va stabili culoarea transparentă pentru o anumită imagine (utilă mai ales pentru generarea fişierelor GIF sau PNG transparente).

int imagecolorstotal (int im) furnizează numărul de culori ale paletei asociate unei imagini.

int imagecolorat (int im, int x, int y) furnizează indexul culorii specifice unui pixel dat de coordonatele *x* şi *y*.

Funcţii pentru utilizarea fonturilor

int imagefontheight (int font) returnează înălţimea în pixeli a unui caracter specificat de fontul respectiv.

int imagefontwidth (int font) returnează lăţimea în pixeli a unui caracter specificat de fontul respectiv.

bool imagepsslantfont (int font_index, double slant) înclină fontul dat de *font_index* cu o înclinaţie dată de parametrul *slant*.

int imagepsloadfont (string filename) încarcă un tip de font PostScript Type 1 dintr-un fişier; dacă totul decurge cu succes va fi returnat un index de font valid care poate fi folosit mai departe în program. În caz contrar, funcţia va returna o valoare invalidă şi va afişa un mesaj de eroare.

```
<?php
// trimitem o imagine JPEG
header ("Content-type: image/jpeg");
$im = ImageCreate (350, 45);
$negru = ImageColorAllocate ($im, 0, 0, 0);
$alb = ImageColorAllocate ($im, 255, 255, 255);
// incercam sa incarcam un font PS
$font = ImagePsLoadFont("bchbi.pfb");
ImagePsText($im, "Corect, merge...",
   $font, 32, $alb, $negru, 32, 32);
// eliberam memoria alocata pentru font
ImagePsFreeFont($font);
// afisam imaginea generata ca JPEG
ImageJpeg($im, "", 100);
ImageDestroy ($im);
?>
```

void imagepsfreefont (int fontindex) eliberează zona de memorie alocată pentru PostScript Type 1.

bool imagepsextendfont (int font_index, double extend) extinde sau comprimă un tip de font în funcţie de argumentul *extend*. Dacă *extend* are o valoare mai mică decât 1, atunci fontul se comprimă.

Alte funcţii utile

array gd_info() oferă informaţii privitoare la biblioteca GD instalată în sistem (este disponibilă doar începând cu versiunea PHP 4.3.0).

int imagesx (int im) şi *int imagesy (int im)* sunt funcţii care furnizează ori stabilesc lăţimea şi respectiv înălţimea unei imagini.

array getimagesize (string filename [, array imageinfo]) returnează dimensiunea unui fişier în format GIF, PNG sau JPEG, plus lăţimea şi înălţimea imaginii conţinute de acesta. Funcţia de fapt nu necesită existenţa bibliotecii GD.

int imagetypes () furnizează ce tipuri de formate grafice sunt suportate de versiunea PHP care rulează pe serverul Web.

```
<?php
if (ImageTypes() & IMG_PNG) {
    // IMG_PNG este o constanta predefinita de PHP
    echo "Suportul pentru PNG este activat.";
}
?>
```

int imagecopy (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h) va copia o zonă dintr-o imagine *src_im* şi o va plasa în cadrul altei imagini (*dst_im*), posibil la alte coordonate.

int imagecopyresized (resource dst_im, resource src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH) funcţionează la fel ca mai sus, dar există posibilitatea modificării şi dimensionării zonei copiate.

int jpeg2wbmp (string jpegname, string wbmpname, int srcH, int srcW, int calitate) poate fi folosită la convertirea unei imagini din format JPEG în format WBMP pentru utilizarea ei de către clienţii WAP.

int png2wbmp (string pngname, string wbmpname, int srcH, int srcW, int calitate) este similară funcţiei precedente, dar necesită ca imagine sursă un fişier în format PNG.

După cum se observă, utilizarea funcţiilor de manipulare grafică puse la dispoziţie de PHP nu prezintă dificultăţi cât timp se respectă cerinţele impuse de folosirea acestora, cum ar fi necesitatea prezenţei altor biblioteci amintite mai sus.

În momentul conceperii unei aplicaţii care intenţionează să folosească funcţii grafice trebuie avute în vedere versiunea de server PHP, precum şi versiunea bibliotecii GD, deoarece există posibilitatea ca unele funcţii să nu funcţioneze din pricina faptului că încă nu au fost incluse în versiunea respectivă. Un exemplu ar fi folosirea funcţiei *imageellipse()*, care a fost adăugată bibliotecii începând cu versiunea 2.0.1, iar pentru PHP începând cu versiunea 4.0.6.; ca urmare utilizarea ei în cazul unor versiuni mai vechi va da, cum era de aşteptat, eroare. Recomandăm astfel consultarea informaţiilor oferite de funcţiile *php_info()* şi *gd_info()*.

exif_imagetype()	imagecreatefromgd2()	imagepalettecopy()
exif_read_data()	imagecreatefromgd2part()	imagepng()
exif_thumbnail()	imagecreatefromgd()	imagepolygon()
gd_info()	imagecreatefromgif()	imagepsbbox()
getimagesize()	imagecreatefromjpeg()	imagepscopyfont()
image_type_to_mime_type()	imagecreatefrompng()	imagepsencodefont()
image2wbmp()	imagecreatefromstring()	imagepsextendfont()
imagealphablending()	imagecreatefromwbmp()	imagepsfreefont()
imagearc()	imagecreatefromxbm()	imagepsloadfont()
imagechar()	imagecreatefromxpm()	imagepsslantfont()
imagecharup()	imagecreatetruecolor()	imagepstext()
imagecolorallocate()	imagedashedline()	imagerectangle()
imagecolorallocatealpha()	imagedestroy()	imagerotate()
imagecolorat()	imageellipse()	imagesetbrush()
imagecolorclosest()	imagefill()	imagesetpixel()
imagecolorclosestalpha()	imagefilledarc()	imagesetstyle()
imagecolorclosesthwb()	imagefilledellipse()	imagesetthickness()
imagecolordeallocate()	imagefilledpolygon()	imagesettile()
imagecolorexact()	imagefilledrectangle()	imagestring()
imagecolorexactalpha()	imagefilltoborder()	imagestringup()
imagecolorresolve()	imagefontheight()	imagesx()
imagecolorresolvealpha()	imagefontwidth()	imagesy()
imagecolorset()	imageftbbox()	imagetruecolortopalette()
imagecolorsforindex()	imagefttext()	imagettfbbox()
imagecolorstotal()	imagegammacorrect()	imagettftext()
imagecolortransparent()	imagegd2()	imagetypes()
imagecopy()	imagegd()	imagewbmp()
imagecopymerge()	imagegif()	iptcembed()
imagecopymergegray()	imageinterlace()	iptcparse()
imagecopyresampled()	imagejpeg()	jpeg2wbmp()
imagecopyresized()	imageline()	png2wbmp()
imagecreate()	imageloadfont()	read_exif_data()

Lista funcţiilor PHP de manipulare a imaginilor

3. Prezentarea aplicaţiei *Dynamic Charts*

Aplicaţia *Dynamic Charts* este un sit Web în care fiecare client are posibilitatea de a crea grafice pentru propriile date, sub formă de *chart*-uri asemănătoare celor din popularele programe de calcul tabelar precum Excel. În plus, datele fiecărui utilizator pot fi memorate într-o bază de date şi pot fi accesate ulterior oricând. O dată introduse datele dorite a fi reprezentate grafic, utilizatorul poate alege tipul de grafic dintr-o listă de grafice predefinite, putând seta şi anumite proprietăţi de reprezentare, precum titlul, etichetele asociate, legenda sau dimensiunea imaginii rezultate.

Partea de desenare a graficelor a fost implementată exclusiv în PHP, folosind o parte a funcţiilor bibliotecii GD. Datele selectate spre a fi figurate au fost stocate în baze de date MySQL.

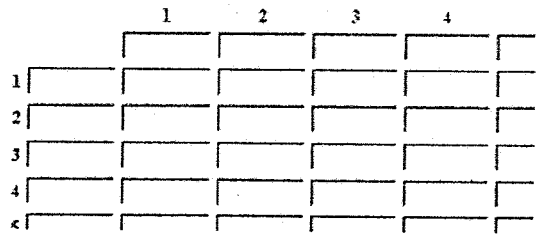
3.1. Utilizarea aplicaţiei

Paşi de urmat

Înainte de a putea utiliza aplicaţia, utilizatorul va trebui să-şi creeze un cont de acces prin intermediul căruia să poată fi ulterior autentificat şi să poată salva datele introduse. Se va recurge la un modul de înregistrare, parola solicitată fiind stocată criptat – prin intermediul funcţiei *md5()* din PHP – în baza de date a utilizatorilor (în tabela denumită *Date_pers*):

```
/* obiect de tip Data - tabela a bazei de date
   care contine informatiile personale */
$db = new Date;
$db->connect(); // conectarea la baza de date
$parola = md5 ($parola); // cripteaza parola
// inserarea datelor despre utilizator
$query = "insert into date_pers " .
  "(nume, prenume, adresa, tara, oras, email, cont,".
  " parola) values ('$nume', '$prenume', '$adresa',".
  " '$tara', '$oras', '$email', '$cont', '$parola')";
$db->query($query); // trimite comanda SQL
```

După introducerea datelor personale, fiecare utilizator îşi poate stoca în baza de date propriile sale date în vederea realizării graficelor dorite. Datele folosite pentru grafice vor fi inserate într-o altă tabelă numită *tabela*. Aceste date vor fi trimise serverului spre prelucrare prin intermediul unui formular ce foloseşte metoda POST. Interfaţa Web are o formă asemănătoare unei foi de calcul:



Interfaţa de introducere a datelor

Codul de creare a formularului XHTML este următorul:

```
<tr>
<?php
```

```
// generarea capului de tabel
for ($i = 1; $i < $nrcol; $i++) {
  echo "<td align='center' bgcolor='#E0F8D8'> $i </td>";
} ?>
</tr>
<?php
$i = 0;
while ($i < $nrlinii) {
?>
<tr class='bg_afis'>
<?php
  if ($i > 0)
    echo "<td align='right' bgcolor='#E0F8D8'> $i </td>";
  else
    echo "<td> </td>";
  $j = 0;
  while ($j < $nrcol) {
      if (($j == 0) && ($i == 0))
         echo "<td> </td>";
      else { // generarea campului de introducere a datelor
         $c = 'nume' . $i . $j;
         echo "<td width='10' height='5'>" .
               "<input name=\"$c\" size='8' /></td>";
      }
      $j++;
  } // while($j)
?>
</tr>
<?php
  $i++; // while($i)
}
?>
```

Din moment ce datele au fost memorate în variabilele *numeij*, cu *i* şi *j* valori întregi, în funcţie de numărul liniilor şi a coloanelor, la apăsarea unui buton de tip *submit* ele vor fi transmise scriptului PHP de inserare în baza de date *insert_aplicant*.

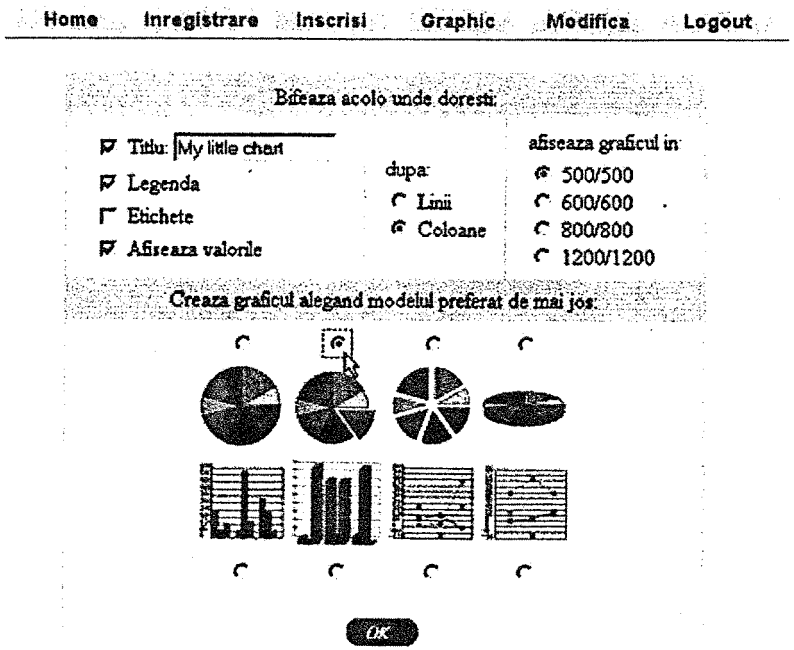
Acest script va introduce într-un tablou toate câmpurile completate în formular:

```
for($i = 0; $i < $nrlinii; $i++) {
  for($j = 0;$j < $nrcol; $j++) {
    $c = 'nume' . $i . $j;
    /* inseram valoarea variabilei al carei nume
       este stocat de variabila $c */
    $v[$j] = $$c;
  }
}
```


Vectorul *$v[$j]* va conţine acum datele, iar ceea ce mai urmează să facem este inserarea lor în baza de date printr-o comandă SQL *insert*. În cazul modificării datelor deja introduse, informaţiile se vor prelucra la fel ca şi cele necesare reprezentării graficului, fără conectarea obligatorie la baza de date.

		1	2	3	4	5	6	7
		Luni	Marti	Miercuri	Joi	Vineri	Sambata	
1	Sapt1	12	24	23	25	15	12	
2	Sapt2	34	-2	78	78	22	33	
3	Sapt3	23	34	23	13	78	22	
4	Sapt4	67	55	56	-5	-43	33	
5								
6								

Introducerea datelor în vederea generării unui *chart*



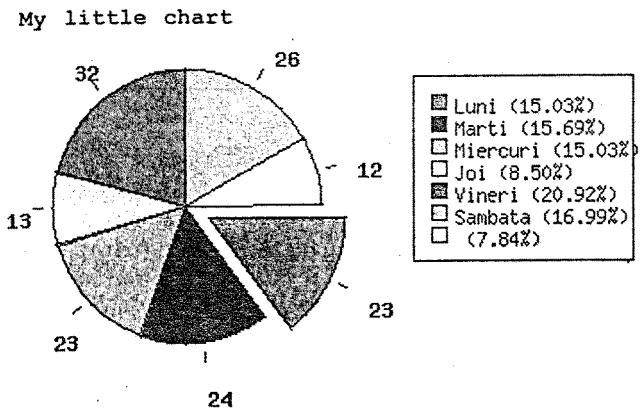
Stabilirea tipului de grafic

Un alt modul - denumit *Înscrişi* - va oferi o listă cu toate persoanele care s-au înscris în baza de date a serverului, în vederea unor eventuale contacte virtuale între utilizatori cu aceleaşi interese.

Pentru realizarea efectivă a graficelor se va recurge la funcţionalitatea modului *Grafic*. Utilizatorul îşi poate introduce datele pentru realizarea graficelor sau, dacă este în contul său, poate folosi datele introduse deja în baza de date, selectându-şi un domeniu de date numerice prin intermediul butoanelor de bifare de tip *checkbox*, ca în captura-ecran anterioară (putem presupune că datele reprezintă vânzările unor produse timp de o lună de zile şi se doreşte realizarea unui grafic comparativ între vânzările din zilele de luni şi de sâmbătă; valorile negative specifică pierderi).

Utilizatorul poate să stabilească anumite proprietăţi referitoare la graficul ce urmează a fi realizat, cum ar fi: dimensiunile în pixeli ale imaginii generate (de exemplu, 800x800), titlul graficului, ataşare de legendă şi etichete, precum şi modul de prezentare a grafului, pe linii sau pe coloane. Poate fi ales şi tipul grafului dorit dintr-o listă de opt implementate.

Presupunând că utilizatorul a făcut alegerile considerate în exemplul de mai sus şi că a folosit datele din tabelul prezentat cu selectarea domeniului respectiv, următorul pas va crea efectiv imaginea grafică, rezultatul putând fi următorul:



Reprezentarea unui grafic de tip plăcintă

3.2. Detalii de implementare

Lucrul cu bazele de date s-a realizat prin intermediul unei clase care încapsulează principalele operaţii ce trebuie realizate: conectarea şi deconectarea de la un server de baze de date, utilizarea unei baze de date, trimiterea de comenzi SQL spre execuţie (de exemplu: *select*, *update* sau *insert*). Modul de implementare a tuturor acestor operaţii este expus în alte capitole ale cărţii de faţă (a se vedea, de exemplu, capitolele 1, 2 şi 7), de aceea nu mai insistăm aici.

Utilizarea funcţiilor bibliotecii GD

Una dintre părţile cele mai importante ale aplicaţiei o reprezintă realizarea concretă a graficelor pentru un set de valori introdus de utilizator la un moment dat. Am folosit pentru aceasta unele funcţii ale bibliotecii GD, funcţii care au fost descrise în cadrul secţiunii 2.1 a capitolului de faţă.

Întreaga implementare este orientată-obiect, pentru fiecare tip de *chart* existând definită o clasă specializată, derivată dintr-o clasă de bază denumită *basechart*. Pentru început o vom prezenta pe aceasta.

Clasa basechart

Clasa generică *basechart* este definită astfel:

```
class basechart {
  // date-membre
  var $im;      // identificatorul pentru imagine
  var $width,
      $height; // latimea si inaltimea imaginii
  var $data;    // valorile de reprezentat in grafic */
  var $colors; // vector de culori pentru imagine
  // latimea si inaltimea graficului va fi calculata
  // in functie de urmatoarele patru variabile
  var $left   = 150;
  var $right  = 200;
  var $top    = 150;
  var $bottom = 180;
  // distanta de sus in jos de la care va fi afisat titlul
  var $head_top = 60;
  // distanta de la stanga la dreapta
  // de unde se va afisa legenda
  var $legend_left = 20;
  var $label       = -1;
  var $legend_num;
  var $fx, $fy; // marimea fonturilor

// metode
// salveaza imaginea intr-un fisier PNG
function display() {
  Imagepng ($this->im, "chart.png");
  Imagedestroy($this->im);
}

// stabileste legenda cu valori in procente
function set_legend_percent (){
```

```
    $this->legend_num = "p";
}

// stabileste daca se face afisarea etichetelor
function set_label($label = 1) {
 $this->label = $label;
}

//stabileste daca se doreste afisarea legendei
function set_legend_value () {
 $this->legend_num = "v";
 $this->fx = array(0, 5, 6, 7, 8 ,9);
 $this->fy = array(0, 7, 8, 10, 14, 19);
}

// adauga titlu la imagine
function add_title($s, $l) {
  $y = $this->head_top;
  $this->black = ImageColorAllocate ($this->im, 0, 0, 0);
  // se verifica unde se doreste afisarea titlului in
  // imagine: la stanga, dreapta sau in mijloc
  switch($l) {
      case "c":
         $x = ($this->width - $this->fy[4]
              * strlen($s)) / 2;
         break;

      case "r": /* folosim marginea stanga... */
         $x = $this->da_width - $this->left -
              ($this->fy[4] * strlen($s));
         break;

      default:
        $x = $this->left;
        break;
  }
  ImageString($this->im, 14, $x, $y, $s, $this->black);
}

// alte metode...
}
```

Tot în cadrul acestei clase sunt incluse metode pentru desenarea legendei şi a etichetelor pentru *chart*-urile figurate:

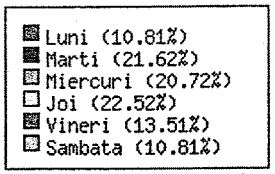

```
function drawlabels ($et) {
/* parametrul $et contine vectorul cu date calculat
   la inceputul programului, fiind prima linie sau
   prima coloana din tabelul de date */
$n = count($et); // numarul de elemente ale lui $et
$sp = ($this->finishx-$this->startx) / $n;
// alocam culoarea neagra de afisare
$black = imagecolorallocate($this->im, 0, 0, 0);
for ($i=1; $i <= $n; $i++) {
   //afiseaza textul etichetei vertical
   ImageStringUp($this->im,3, $this->startx + $i *
      $sp - $sp / 2, $this->starty + 7 *
      strlen ($et[$i-1]), $et[$i-1], $black);
  } // for
}
```

Pentru crearea legendei în cazul graficului de tip *Pie* am creat următoarea metodă:

```
function draw_pielegends ($fontsize, $leg, $v) {
// $leg reprezinta vectorul cu textele de inserat
// in legenda, iar $v vectorul datelor numerice
$n = count ($leg); // numarul de legende figurate
// calculam latimea maxima a unui text din legenda
$maxcuv = strlen($leg[0]);
for ($i=1; $i < $n; $i++)
  if ($maxcuv < strlen($leg[$i]))
    $maxcuv = strlen($leg[$i]);
// calculam coordonatele de afisare (intai X)
$x1 = $this->width - $this->right +
      3 * $this->legend_left;
$x2 = $x1 + $this->fy[$fontsize];
// iteram vectorul $leg pentru a afisa legendele
for ($k=0; $k < $n; $k++) {
   // alegem o culoare de afisare
   $this->color[$k] = ImageColorAllocate($this->im,
     ($k-1)*255,($k-1)*100,($n-$k)*100);
}
for ($i = 0; $i < $n; $i++) {
   // determinam coordonata Y
   $y1 = ($i * $this->fy[$fontsize] * 1.5) +
          $this->da_height / 2 +
          $this->top-$this->head_top;
   $y2 = $y1 + $this->fy[$fontsize];

   // desenam dreptunghiul colorat al legendei
```

```
    ImageFilledRectangle($this->im,
       $x1, $y1, $x2, $y2, $this->color[$i]);
    ImageRectangle($this->im, $x1, $y1, $x2, $y2,
       $this->black);

    // cream sirul de afisat ca legenda,
    // dupa dreptunghiul colorat
    $legend = $leg[$i];
    $legend .= sprintf(" (%.2f%%)",
                  ($v[$i] * 100 / doubleval($this->sum)));

    // afisam sirul
    ImageString($this->im, $fontsize, $x2 + 5, $y1,
       $legend, $this->black);
}
// stabilim coordonatele chenarului legendei
$xls = $x1 - 10;
$xle = $x2 + 6 * $maxcuv + 3 * strlen($legend) + 30;
$yls = $this->da_height / 2 +
       $this->top-$this->head_top - 10;
$yle = $y2 + 10;
// desenam chenarul legendei
ImageLine($this->im, $xls, $yls, $xls, $yle,$this->black);
ImageLine($this->im, $xls, $yls, $xle, $yls,$this->black);
ImageLine($this->im, $xle, $yls, $xle, $yle,$this->black);
ImageLine($this->im, $xle, $yle, $xls, $yle,$this->black);
}
```

De exemplu, am putea avea următorul rezultat:



Exemplu de generare a legendei unui grafic

În mod similar este realizată şi metoda pentru afişarea legendei în cazul graficelor de tip bară, cu deosebirea că vor lipsi din textul afişat valorile datelor în procente.

În continuare vom prezenta o parte dintre clasele mai complexe pentru crearea unor tipuri de grafice, celelalte clase având o structură asemănătoare, astfel încât cititorul poate încerca – drept exerciţiu util – să le implementeze de unul singur.

Clasa barchart3D

Clasa *barchart3D* este clasa pentru crearea unui grafic de tip *barchart* în spaţiu. La crearea unui obiect de tipul acestei clase se va apela automat constructorul clasei (cu acelaşi nume), în care se generează o imagine de dimensiuni specificate, se alocă memorie pentru culorile folosite în reprezentarea fiecărei date precum şi a fundalului şi se calculează dimensiunile graficului încadrat în imaginea creată. Culorile vor avea valori în funcţie de numărul de date ce vor fi reprezentate:

```
// fundal alb
$background_color = ImageColorAllocate
                        ($this->im, 255, 255, 255);
for($k = 0; $k < $this->nr_bar; $k++)
  $this->color[$k] = ImageColorAllocate($this->im,
     ($k-1) * 255, ($k-1) * 100,
     ($this->nr_bar - $k) * 100);
$this->black= ImageColorAllocate ($this->im, 0, 0, 0);
```

În funcţia de desenare a graficului calculăm valoarea maximă şi pe cea minimă din vectorul de valori ce urmează a fi reprezentate, în vederea translaţiei originii, a reprezentării axelor şi a valorilor de pe axe şi pentru calcularea înălţimii maxime pe care o poate avea un *barchart*.

Tot în funcţie de numărul datelor se calculează distanţele dintre barele figurate şi lăţimea lor. În funcţie de aceste date calculate am folosit funcţiile de desenare pentru trasarea axelor şi a fiecărui *barchart* în parte (omitem o parte dintre detaliile de implementare):

```
// metoda de desenare a unei bare a graficului barchart
function drawbar() {
  $n = count($this->data[0]);
  $max = max($this->data[0]);
  $min = min($this->data[0]);
  $pas = $this->da_width /(($this->nr_bar) * $n);
  /* spatiul dintre barchart-uri */
  $sp = $pas / 4;
  /* latimea unui barchart */
  $pasw = $pas - $sp;
  $mar = $pas / 2;
  for($k=1; $k<$this->nr_bar; $k++) {
    $maxp = max($this->data[$k]);
    $minp = min($this->data[$k]);
    $max  = max($max,$maxp);
    $min  = min($min,$minp);
  }
  $scade = 0;
```

```
if($min < 0) {
      $scade = $this->da_height / ($max - $min);
      $plus = $scade * $max;
      $scade = $scade * abs($min);
}
$y = $this->starty-$scade;
if($max > abs($min)) {
      $q = $this->numar($max);
      /* numar este o functie in care am calculat
         ordinul numarului maxim (ex: 10, 100...) */
      $q = (intval($max / $q) + 1) * $q;
      $p = ($this->da_height - $scade) / 10;
      $pash = ($this->da_height - $scade) / $q;
}
else {
      $q = $this->numar(abs($min));
      $q = (intval(abs($min) / $q) + 1) * $q;
      $p = $scade / 10;
      $pash = $scade / $q;
} // if
// desenam axele si valorile de pe axe
$i = 0;
while($y > $this->finishy) {
   ImageString($this->im, 2, $this->startx - 15,
                $y - 10, $i, $this->black);
   $i = $i + $q / 10;
   $y = $y - $p;
   ImageLine($this->im, $this->startx + $mar,
             $y - $mar, $this->finishx + $sp + $mar,
             $y - $mar, $this->black);
   ImageLine($this->im, $this->startx, $y,
             $this->startx + $mar, $y - $mar,
             $this->black);
} // while
ImageString($this->im, 2, $this->startx - 15,
            $y - 10, $i, $this->black);
ImageLine($this->im, $this->startx, $y,
          $this->startx + $mar, $y - $mar,
          $this->black);
ImageLine($this->im, $this->startx + $mar, $y - $mar,
          $this->finishx + $sp + $mar, $y - $mar,
          $this->black);
ImageLine($this->im, $this->startx,
          $this->starty - $scade,
```



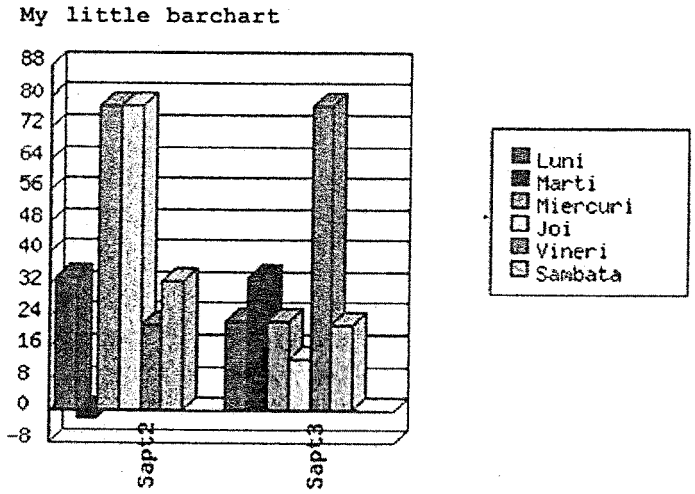
```
          $this->startx + $mar,
          $this->starty - $scade - $mar, $this->black);
ImageLine($this->im, $this->startx,
          $this->starty - $scade,
          $this->finishx + $sp,
          $this->starty - $scade, $this->black);
ImageLine($this->im, $this->startx + $mar,
          $this->starty - $scade - $mar,
          $this->finishx + $sp + $mar,
          $this->starty - $scade - $mar, $this->black);
ImageLine($this->im, $this->finishx + $sp,
          $this->starty - $scade,
          $this->finishx + $sp + $mar,
          $this->starty - $scade - $mar, $this->black);
/* daca valorile sunt negative,
   desenarea lor si a axelor se vor face in jos */
if ($min < 0) {
   $z = $this->starty - $scade + $p;
   $j = -$q / 10;
   while($z <= $this->starty) {
     ImageLine($this->im, $this->startx + $mar,
               $z - $mar, $this->finishx + $sp + $mar,
               $z - $mar, $this->black);
     ImageLine($this->im, $this->startx, $z,
               $this-> startx + $mar, $z - $mar,
               $this->black);
     ImageString($this->im, 2, $this->startx - 20,
                 $z - 10, $j, $this->black);
     $j = $j - $q / 10;
     $z += $p;
   } // while

   ImageLine($this->im, $this->startx, $z,
             $this->finishx + $sp, $z, $this->black);
   ImageString($this->im, 2, $this->startx - 20,
               $z - 10, $j, $this->black);
}
else
  $z = $this->starty;
ImageLine($this->im, $this->startx, $z, $this->startx,
          $y, $this->black);
ImageLine($this->im, $this->finishx + $sp + $mar,
          $z - $mar, $this->finishx + $sp + $mar,
          $y - $mar, $this->black);
```

```
ImageLine($this->im, $this->finishx + $sp + $mar,
          $z - $mar, $this->startx + $mar, $z - $mar,
          $this->black);
ImageLine($this->im, $this->startx, $z,
          $this->startx + $mar, $z - $mar,
          $this->black);
ImageLine($this->im, $this->startx + $mar, $z - $mar,
          $this-> startx + $mar, $y - $mar,
          $this->black);
ImageLine($this->im, $this->finishx + $sp + $mar,
          $z - $mar, $this->finishx+$sp, $z,
          $this->black);

/* aici incepe desenarea fiecarui barchart */
$xf = 0;
for ($k = 0; $k < $this->nr_bar; $k++) {
   $xs = $this->startx + ($k) * $pasw + $sp;
   $yf = $this->starty - $scade;
   for ($i = 0; $i < $n; $i++) {
      $ys = $yf - $this->data[$k][$i] * $pash;
      $xf = $xs + $pasw;
   /* daca datele sunt pozitive le vom desena
      in partea pozitiva a axelor,
      altfel in cea negativa */
      if ($this->data[$k][$i] > 0) {
         Imagefilledrectangle($this->im, $xs, $ys,
            $xf, $yf, $this->color[$k]);
         // simulam si caracterul 3D al barelor
         ImageLine($this->im, $xs, $ys,
            $xs + $mar, $ys - $mar, $this->color[$k]);
         ImageLine($this->im, $xs + $mar, $ys - $mar,
            $xf + $mar, $ys - $mar, $this->color[$k]);
         ImageLine($this->im, $xf + $mar, $ys - $mar,
            $xf, $ys, $this->color[$k]);
         ImageLine($this->im, $xf, $ys, $xs, $ys,
            $this->color[$k]);
         ImageLine($this->im, $xf + $mar, $ys - $mar,
            $xf + $mar, $yf - $mar, $this->color[$k]);
         ImageLine($this->im, $xf + $mar, $yf - $mar,
            $xf, $yf, $this->color[$k]);
         ImageFillToBorder($this->im,
            ($xs + $xf) / 2 + 2, $ys - 2,
             $this->color[$k], $this->color[$k]);
```



```
         ImageFillToBorder($this->im, $xf + 2,
            ($ys + $yf) / 2 - 2,
             $this->color[$k], $this->color[$k]);
      } // if (similar si pe ramura else...)
      $xs = $xf + ($this->nr_bar - 1) * $pasw +
            ($this->nr_bar) * $sp;
   } // for ($i)
} // for ($k)
} // final al metodei
```

Prezentăm un exemplu de grafic *barchart* afişat împreună cu etichete şi legendă, construit cu date din acelaşi tabel de date dat mai sus:



Un grafic Bar 3D generat de aplicaţia *Dynamic Charts*

Clasa barchart

Clasa pentru crearea unui grafic de bare simplu este asemănătoare cu clasa pentru crearea lui în spaţiu, însă lipsesc din construcţia acestuia funcţiile de desenare a părţilor care simulează aspectul tridimensional al barelor figurate.

Clasa pointchart

Şi această clasă este similară clasei *barchart3D*, dar în locul barelor desenăm puncte dreptunghiulare – prin intermediul funcţiei GD *imagefilledrectangle()*. Repartizarea lor nu se mai realizează prin clădire pe aceeaşi axă verticală. Funcţia de desenare conţine următoarele linii de cod:

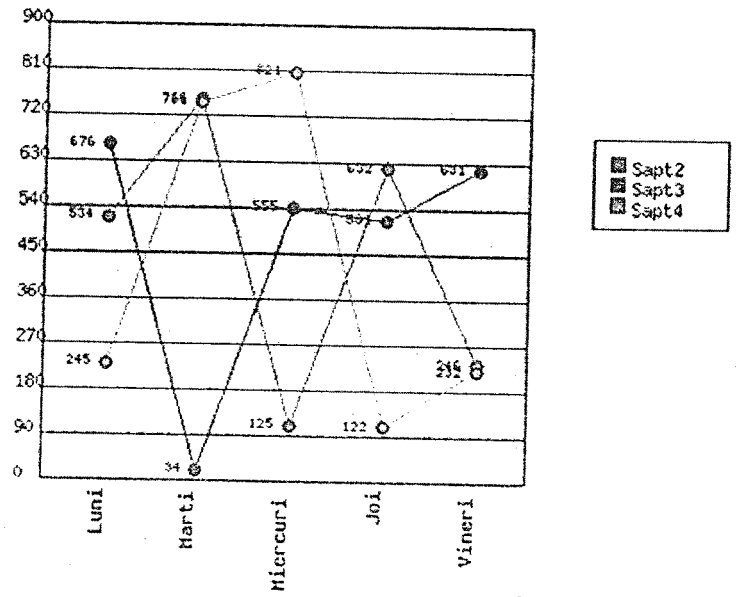
```
for ($k = 0; $k < $this->nr_bar; $k++) {
   $xd = 0;
```

```
    $yd = 0;
    $xf = $this->startx;
    $yf = $this->starty - $scade;
    for ($i = 0; $i < $n; $i++) {
        $xs = $xf + $sp;
        $ys = $yf - $this->data[$k][$i] * $pash;
        $xf = $xs + $pasw;
        $xp = $xd;
        $yp = $yd;
        if ($this->data[0][$i] > 0) {
          // figurarea punctelor
          Imagefilledrectangle($this->im, ($xs + $xf) / 2,
             $ys, ($xs + $xf) / 2 + $mar, $ys + $mar,
             $this->color[$k]);
          Imagerectangle($this->im, ($xs + $xf) / 2, $ys,
            ($xs + $xf) / 2 + $mar, $ys + $mar,
             $this->black);
          $xd = (($xs + $xf) / 2 + ($xs + $xf) / 2 +
                  $mar) / 2;
          $yd = ($ys + $ys + $mar) / 2;
        }
        else {
          Imagefilledrectangle($this->im, ($xs + $xf) / 2,
            $ys - $mar, ($xs + $xf) / 2 + $mar, $ys,
            $this->color[$k]);
          Imagerectangle($this->im, ($xs + $xf) / 2,
            $ys - $mar, ($xs + $xf) / 2 + $mar, $ys,
            $this->black);
          $xd = (($xs + $xf) / 2 + ($xs + $xf) / 2 +
                  $mar) / 2;
          $yd = ($ys - $mar + $ys) / 2;
        } // if
        if ($xp != 0 && $yp != 0)
          ImageLine($this->im, $xp, $yp, $xd, $yd,
             $this->color[$k]);
    } // for
}
```

Clasa circlepointchart

Această clasă este asemănătoare clasei *pointchart*, singura deosebire fiind înlocuirea dreptunghiului colorat cu un mic cerc, realizat prin funcţia *imagearc()*.

Prezentăm mai jos un exemplu concludent de generare a unui astfel de grafic:



Exemplu de grafic de tip *circlepointchart*

Clasa piechart

Crearea fiecărui tip de grafic *piechart* oferit de *Dynamic Charts* se realizează prin instanţierea unei aceleiaşi clase – denumită *piechart* – care conţine metode specifice fiecărui tip dorit de grafic sub formă de plăcintă. Ca date membre ale acestei clase pot fi menţionate următoarele:

```
// coordonatele centrului placintei figurate
var $center_x;
var $center_y;
// diametrul placintei
var $diameter;
// valoarea care indica daca va avea sau nu loc
// desfacerea (explode) a unui sector din piechart
var $expl = -1;
// variabila pentru afisarea etichetelor
var $label = -1;
/* suma valorilor datelor */
var $sum;
/* marimile fonturilor folosite */
var $fx, $fy;
// definirea unui PI suficient de exact
var $PI = 3.1415926535897931;
```

Constructorul acestei clase este similar cu cel prezentat la clasa *barchart3D*. Constructorul actual calculează dimensiunile graficului din imagine, desemnează culorile folosite şi, în plus, setează centrul de poziţionare a *piechart*-ului în imagine.

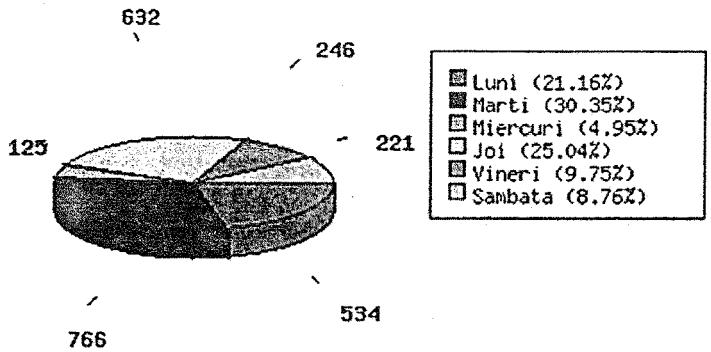
Desenarea efectivă a graficului de acest tip se realizează prin intermediul metodelor *drawpie()*, *drawexplosionpie()* şi *drawpie3D()*, metode în care sunt folosite – după cerinţe şi în funcţie de numărul de date ce urmează a fi reprezentate – două funcţii de desenare a fiecărui *slice* (un sector de cerc corespunzător unei felii din plăcinta figurată). Aceste funcţii au fost denumite *draw_slice()* şi *draw_slice3D()*.

```
function draw_slice ($x, $y, $from, $to, $color,
                     $la, $j, $cal) {
  $colorb = ImageColorAllocate($this->im,   0,   0,   0);
  // deseneaza un arc intre punctele specificate
  // unghiular prin $from si $to
  ImageArc($this->im, $x, $y, $this->diameter,
           $cal, $from, $to, $colorb);
  // transformam in radiani
  $axy2 = $this->get_xy_factors($from);
  // calculam punctele carteziene
  $ax2 = floor($x + ($axy2[0] * ($this->diameter / 2)));
  $ay2 = floor($y + ($axy2[1] * ($cal / 2)));
  // desenam prima linie din cadrul slice-ului
  ImageLine($this->im, $x, $y, $ax2, $ay2, $colorb);
  $bxy2 = $this->get_xy_factors($to);
  $bx2 = ceil($x + ($bxy2[0] * ($this->diameter / 2)));
  $by2 = ceil($y + ($bxy2[1] * ($cal / 2)));
  // ... si a doua linie
  ImageLine($this->im, $x, $y, $bx2, $by2, $colorb);
  $xy2 = $this->get_xy_factors((($to - $from) / 2) + $from);
  $x2 = floor($x + ($xy2[0] * ($this->diameter / 3)));
  $y2 = floor($y + ($xy2[1] * ($cal / 3)));
  // umplem slice-ul desenat cu culoarea specifica datei
  ImageFillToBorder($this->im, $x2, $y2, $colorb, $color);
  // daca este setata valoarea pentru afisarea valorilor,
  // se face desenarea acestora
  if ($this->label != -1) {
    $x3 = floor($x + ($xy2[0] * ($this->diameter / 2 + 10)));
    $y3 = floor($y + ($xy2[1] * ($this->diameter / 2 + 10)));
    $x4 = floor($x + ($xy2[0] * ($this->diameter - 70)));
    $y4 = floor($y + ($xy2[1] * ($this->diameter - 70)));
    $x5 = floor($x + ($xy2[0] * ($this->diameter - 55 +
              2 * strlen($this->data[$j]))));
    $y5 = floor($y + ($xy2[1] * ($this->diameter - 55 +
              2 * strlen($this->data[$j]))));
```



```
    $x6 = floor($x + ($xy2[0] * ($this->diameter - 35)));
    $y6 = floor($y + ($xy2[1] * ($this->diameter - 35)));
    if($this->label == 1) {
      ImageLine($this->im, $x3, $y3, $x4, $y4, $colorb);
      ImageString($this->im, 2, $x6, $y6, $this->data[$j],
                  $colorb);
    }
    else
       if($this->label == 2) {
          ImageLine($this->im, $x3, $y3, $x4, $y4, $colorb);
       ImageString($this->im, 3, $x5, $y5,
                   $this->data[$j], $colorb);
       }
    } // if
  } // if
}
```

Pentru al doilea tip de *piechart* pus la dispoziţie se apelează metoda *explode()* care va conduce la setarea unei variabile *expl* testată în metoda *drawpie()* astfel încât desenarea unui sector de cerc să se realizeze la o anumită distanţă de centru.



Captură-ecran al unui grafic de tip plăcintă 3D

Metoda *drawexplosionpie()* detaşează toate feliile de plăcintă ale graficului, nu doar una singură.

Ultimul tip de grafic *piechart* este de fapt o variantă tridimensională a graficului de tip plăcintă. Acesta se construieşte asemănător celorlalte tipuri de *piechart*-uri, apelându-se metoda *drawpie3D()*. Metoda va folosi o funcţie *draw_slice3D()* care realizează o translaţie cu o anumită valoare a absciselor punctelor utilizate în construcţie, cum ar fi centrul plăcintei şi punctele unghiulare.

4. În loc de concluzii

După cum s-a putut remarca din cele descrise mai sus, nu există dificultăţi majore în folosirea funcţiilor grafice ale bibliotecii GD sub PHP. Singura dificultate a fost calcularea coordonatelor punctelor primitivelor grafice utilizate de aceste funcţii, o activitate suficient de laborioasă. Aplicaţia *Dynamic Charts* a necesitat multă atenţie la partea de desenare a fiecărui tip de *chart*, în mod special la umplerea unor zone cu anumite culori. Cititorul este provocat să îmbunătăţească modul de figurare a graficelor, în vederea unei mai bune poziţionări a textelor din cadrul etichetelor, legendelor şi valorilor. De asemenea, se pot implementa şi alte tipuri de grafice (*Area*, *Bubble*, *Cylinder*, *Cone* etc.), prin scrierea unor clase specifice care ulterior vor putea fi folosite în cadrul aplicaţiei Web.

Pe viitor ar putea fi oferite utilizatorilor unele facilităţi, precum alegerea de fonturi pentru scrierea titlului, a legendei şi etichetelor, setarea unei anumite culori pentru fundalul imaginilor sau alocarea culorilor independent de datele folosite.

Sitului Web al aplicaţiei i se poate adăuga şi o galerie de *chart*-uri create pe exemple concrete, ceea ce ar oferi utilizatorului nou-venit o viziune mai cuprinzătoare asupra facilităţilor pe care i le pune la dispoziţie *Dynamic Charts*.

Comentarii bibliografice

Un punct de început în redactarea acestui capitol a fost articolul S. Buraga, „Manipularea dinamică a imaginilor în PHP", apărut în numărul 110 (noiembrie 2001) al revistei *NET Report*. Acest articol e disponibil pe Web la **http://www.infoiasi.ro/~busaco/publications/articles/php_graph.pdf**.

De asemenea, se poate consulta şi cartea *Web Application Development with PHP 4.0*, scrisă de T. Ratschiller şi T. Gerken, apărută la New Riders Publishing în anul 2000.

Cele mai recente versiuni ale bibliotecilor menţionate în cadrul celei de a doua secţiuni a capitolului sunt disponibile pe Web la următoarele adrese (aceste biblioteci pot fi instalate mai ales pe un sistem Linux):

- Biblioteca *GD*: **http://www.boutell.com/gd/**
- Biblioteca *jpeg*: **ftp://ftp.uu.net/graphics/jpeg/**
- Biblioteca *libpng*: **http://www.libpng.org/pub/png/libpng.html**
- Biblioteca pentru fonturi *FreeType*: **http://www.freetype.org/**
- Biblioteca pentru fonturi *PS Type 1*: **ftp://sunsite.unc.edu/pub/Linux/libs/graphics/**
- Biblioteca *xpm*: **ftp://metalab.unc.edu/pub/Linux/libs/X/!INDEX.html**

Alte adrese utile, demne de interes, sunt **http://www.weblogs.com/GD/** şi **http://www.phpbuilder.com/**.

Alte amănunte privitoare la funcţiile de manipulare grafică oferite de PHP pot fi găsite chiar în manual (reamintim că o variantă *on-line* este disponibilă la adresa **http://www.php.net/docs.php**).

CAPITOLUL 7

Accesul la baze de date prin servicii Web

Silvana Solomon, sylvy@infoiasi.ro

Acest capitol prezintă principalele trăsături ale protocolului XML-RPC, prin care e posibilă apelarea de proceduri la distanţă, mesajele schimbate fiind scrise în format XML, în vederea implementării de servicii Web bazate pe XML. Exemplificăm utilizarea tehnologiei XML-RPC pentru gestiunea unei baze de date Web indiferent de platformele folosite.

1. O introducere în servicii Web

De câţiva ani, lumea informaticii a evoluat destul de mult în ceea ce priveşte utilizarea de aplicaţii distribuite, disponibile pe calculatoare localizate oriunde pe mapamond, principala activitate fiind cea de *comunicare*.

Interfeţele software utilizate pentru înlesnirea comunicării între diverse situri sunt referite prin termenul generic de *servicii Web*. Deşi există mai multe definiţii ale acestui concept, un serviciu Web poate fi considerat ca fiind un sistem software proiectat pentru a suporta interacţiunea maşină-maşină la nivel de Internet, având o interfaţă descrisă într-un format procesabil de către maşină. Conform lui Graham Glass, un serviciu Web reprezintă „o colecţie de funcţii împachetate, văzute ca o singură entitate şi publicate pe reţea cu scopul de a fi folosite de alte programe; serviciile Web sunt cărămizile pentru crearea de sisteme distribuite deschise, permiţând companiilor şi indivizilor să-şi facă disponibile pe Web bunurile digitale într-un mod rapid şi facil".

Din această definiţie se poate întrevedea faptul că serviciile Web reprezintă o standardizare a modului de realizare a afacerilor pe Web. Astfel, serviciile Web formează un cadru mai general pentru înlesnirea modului de comunicare pe reţea (şi, totodată, de interconectare a componentelor programabile ale Web-ului), având la bază ideea atomică de serviciu. O astfel de standardizare este posibilă cu concursul altor standarde deja existente şi utilizate pe scară largă, dintre care cel mai important este XML. O altă cauză favorabilă o reprezintă consolidarea protocolului HTTP pentru transportul textului – şi altor tipuri de conţinut MIME – pe reţea.

1.1. Standarde

Descrierea unui serviciu Web se realizează în limbajul *WSDL* (*Web Service Description Language*), care oferă un model şi o sintaxă bazate pe XML pentru aceasta. WSDL permite separarea descrierii funcţionalităţii abstracte oferite de un serviciu de detaliile concrete ale descrierii acelui serviciu (de tipul „cum" şi „unde" este pusă la dispoziţie acea funcţionalitate).

Limbajul WSDL descrie serviciile Web începând cu mesajele care sunt schimbate între entitatea care oferă şi cea care invocă un anumit serviciu. Mesajele sunt descrise în manieră abstractă şi apoi sunt asociate unui protocol de reţea, precizându-se de asemenea şi un format (o sintaxă). Un mesaj constă dintr-o colecţie de date de anumite tipuri, iar schimbul de mesaje este descris ca o operaţie.

În contextul actual al spaţiului World-Wide Web, existenţa serviciilor Web bazate pe XML este deosebit de importantă, mai ales în ceea ce priveşte integrarea aplicaţiilor şi schimbul de informaţii între partenerii de afaceri – *business-to-business*. De exemplu, o companie poate oferi un serviciu Web de facturare pe care furnizorii companiei îl folosesc pentru a trimite facturi electronice. Analog, un vânzător ar putea pune la dispoziţie un serviciu Web pentru efectuarea electronică a comenzilor.

Apare însă o problemă. Urmărind scenariul de mai sus, la un moment dat compania amintită ar dori să cumpere – via mijloace electronice – diverse echipamente hardware, trebuind să caute toţi vânzătorii care comercializează în acest mod astfel de echipamente. Pentru aceasta, compania ar avea nevoie de un catalog tip *Pagini Aurii* al tuturor furnizorilor de servicii Web în vederea regăsirii informaţiilor dorite. Acest catalog există şi este numit *UDDI* (*Universal Description, Discovery, and Integration*). UDDI a luat naştere în anul 2000 la iniţiativa unor corporaţii precum Ariba, IBM, Microsoft şi alte 33 de companii, actualmente comunitatea UDDI cuprinzând peste 200 de membri.

Ca şi un catalog clasic, UDDI oferă o bază de date a afacerilor, acestea putând fi căutate după tipul afacerii ori alte criterii. Uzual, căutarea se realizează în funcţie de clasa din care face parte afacerea, după un nume de serviciu sau după locaţia geografică a afacerii respective.

Fiecare afacere înregistrată via UDDI îşi prezintă toate serviciile şi atribuie fiecăruia dintre aceste servicii un tip. Un tip de serviciu îşi are originea într-o mulţime de tipuri de servicii bine cunoscute care sunt înregistrate cu UDDI. Tipurile de servicii sunt numite *tModele* în limbajul UDDI. Fiecare *tModel* are un nume, o descriere şi un identificator unic, numit *tModelKey*. Având la dispoziţie o mulţime de tipuri de servicii bine cunoscute, UDDI dă posibilitatea de a afla modalitatea prin care se pot iniţia afaceri electronice cu o companie. Acesta este principalul avantaj pe care îl are UDDI în comparaţie cu alte cataloage de afaceri bazate pe Web.

Rezumând, tehnologiile pe care se bazează serviciile Web sunt:

- o metodă folosită pentru înregistrarea şi ulterior regăsirea serviciilor Web, urmând principiile proiectării serviciilor de nume şi de directoare. Standardul folosit în prezent este UDDI;

- o manieră standardizată de descriere a serviciilor Web, denumită WSDL;
- un mecanism de apelare (invocare) a serviciilor Web – protocolul de comunicare –, permiţând transmisia în format XML a parametrilor şi întoarcerea răspunsului. Ca exemple se pot menţiona SOAP şi XML-RPC.

Peste protocolul de comunicare poate fiinţa un alt protocol specific, de exemplu *ebXML* folosit pentru conversaţii de afaceri şi reprezentând o metodă standardizată pentru modelarea de afaceri B2B prin intermediul serviciilor Web.

1.2. RPC şi XML-RPC

Protocolul prin intermediul căruia au loc schimburile de mesaje între un ofertant de servicii Web şi clienţii săi urmează modelul clasic folosit de aplicaţiile distribuite. Pentru a înţelege maniera de interacţiune dintre diverse servicii Web şi alte aplicaţii, vom descrie pentru început apelul de proceduri la distanţă.

Pe un calculator izolat, aplicaţiile folosesc apelurile de proceduri pentru a-şi realiza sarcinile. Oricare program (indiferent de limbajul în care e scris) implementează diverse proceduri (subrutine, funcţii, metode etc.) pentru realizarea unor sarcini specifice (de la simpla sortare a unei liste până la imprimarea unui document). Acelaşi tip de serviciu este cerut pentru a folosi capacităţile accesibile de pe alte calculatoare, ideea fiind aceea de a beneficia de funcţionalităţile implementate de proceduri (eventual concepute în alte limbaje de programare) incluse în programe aflate la distanţă. Paradigma folosită este cea a *apelurilor de proceduri la distanţă* (*Remote Procedure Call – RPC*).

Privire de ansamblu

Paradigma RPC reprezintă o extensie foarte simplă a ideii apelului de procedură şi a modelului client/server. Pur şi simplu se creează conexiuni între procedurile care se execută în aplicaţii diferite sau pe maşini diferite. Programul care apelează o procedură la distanţă este considerat client, iar serverul RPC va oferi diverse servicii de apelare a acestor proceduri şi de trimitere către client a rezultatelor. Avantajul adus de o abordare RPC între platforme este că permite, de exemplu, sistemelor UNIX să comunice cu cele Windows şi invers. Dezavantajul este că schimbul de date între maşini se realizează la nivel binar, prin intermediul unor codificări speciale *XDR* (*External Data Representation*). Pentru mai multe amănunte, cititorul poate consulta capitolul 11 al cărţii *Atelier de programare în reţele de calculatoare*, de Sabin Buraga şi Gabriel Ciobanu. Drept „nepoţi" ai mecanismului RPC se pot menţiona paradigmele obiectuale de realizare a aplicaţiilor distribuite: OMG CORBA şi Microsoft COM.

Pentru a înlătura dezavantajul schimbului binar de mesaje, s-au formulat diverse propuneri de codificare a datelor într-un format universal, standardizat, text. Soluţia a fost aceea de a adopta limbajul XML pentru interschimbul de date între clienţi şi servere RPC. Astfel argumentele unei proceduri (metode) apelate (sau invocate) la distanţă sunt codificate via un protocol bazat pe XML, la fel şi răspunsul returnat.

În fapt, aceste proceduri apelate la distanţă reprezintă funcţionalităţi puse la dispoziţie de serviciile Web.

XML-RPC oferă o specificaţie şi un set de implementări care permit software-ului care rulează pe sisteme de operare dispersate şi în medii eterogene să realizeze apeluri de proceduri prin Internet. Foloseşte protocolul *HTTP* (*HyperText Transfer Protocol*) pentru transportul efectiv al mesajelor şi limbajul *XML* (*Extensible Markup Language*) pentru codarea acestora. Este proiectat pentru a fi cât mai simplu, în acelaşi timp însă permiţând ca structuri complexe de date să fie transmise, procesate şi returnate.

În XML-RPC pasarea şi returnarea structurilor complexe de date este facilă: tablourile şi structurile (*struct*) pot fi folosite la fel de simplu ca şi tipurile de date primitive.

Până acum există implementări cunoscute în mai multe limbaje de programare. Sunt disponibile servere pentru PHP, Frontier, Java, Perl, Tcl şi Zope. Implementări ale clientului există pentru limbaje sau paradigme ca PHP, Frontier, COM, Perl, TCL şi Python.

XML-RPC vs. *SOAP*

Un concurent al protocolului XML-RPC este *SOAP* (*Simple Object Access Protocol*). Acesta este, de asemenea, un protocol pentru schimbul de informaţii într-un mediu descentralizat şi distribuit. Este un protocol bazat pe XML alcătuit din trei părţi: un plic care defineşte cadrul de lucru pentru a descrie ceea ce se află într-un mesaj şi modul de a-l procesa, un set de reguli de codare pentru a exprima instanţe ale tipurilor de date definite în aplicaţie şi o convenţie pentru a reprezenta apelurile de proceduri la distanţă şi răspunsurile.

În timp ce XML-RPC intenţionează doar să realizeze apelurile de proceduri la distanţă la un nivel simplu, dar păstrând un format necomplicat, protocolul SOAP propune facilităţi sofisticate şi oferă mult mai multe posibilităţi de reprezentare (serializare şi deserializare) a datelor vehiculate.

Cea mai importantă trăsătură a XML-RPC este simplitatea, în timp ce SOAP, deşi oferă mai multe utilităţi, este mai puţin natural în utilizare.

Dacă este nevoie de tipuri de date complexe definite de utilizator şi posibilitatea ca fiecare mesaj să definească cum va fi procesat, atunci SOAP este soluţia mai bună. Totuşi, dacă tipurile de date standard şi apelurile simple de metode sunt suficiente, atunci XML-RPC face aplicaţiile mai rapide şi mai uşor de depanat şi întreţinut.

Indiferent de protocolul utilizat – XML-RPC ori SOAP –, avantajul major este acela că există posibilitatea realizării de apeluri de proceduri la distanţă folosind direct tehnologiile Web, fără a trebui concepute aplicaţii sofisticate. Astfel, Web-ul poate fi considerat mediul cel mai flexibil şi uşor de utilizat pentru realizarea de aplicaţii distribuite complexe. În jargonul specialiştilor, procedurile care pot fi apelate la distanţă se numesc *servicii Web*, actualmente existând cel puţin două platforme (şi filosofii) pentru implementarea şi exploatarea de servicii Web sofisticate: *ONE* (*Open Network Environment*) de la Sun şi *.NET* de la Microsoft.

În continuare vom vedea cum putem implementa servicii Web simple direct în PHP prin utilizarea facilităţilor puse la dispoziţie de XML-RPC.

2. XML-RPC pentru PHP

În cadrul acestui subcapitol vom prezenta o soluţie practică de implementare orientată-obiect a unor programe PHP pentru transmiterea de mesaje codificate conform specificaţiilor XML-RPC. Astfel, folosim o colecţie de clase PHP care oferă un cadru de lucru pentru scrierea clienţilor şi serverelor XML-RPC în PHP. Aceste clase vor fi ulterior utilizate pentru a dezvolta o aplicaţie de interogare de la distanţă a unor servere de baze de date (de exemplu, MySQL ori PostgreSQL). Bazele pot fi stocate pe o platformă (de exemplu, Windows), iar clienţii care le interoghează pe o alta (să presupunem, de pildă, un sistem Linux). Desigur, acest schimb de informaţii se poate realiza şi direct, folosind scripturi CGI ori servere de aplicaţii ca PHP, ASP sau JSP, dar rezultatul primit de la server ar fi fost o simplă pagină Web (marcată în HTML) care ar fi fost ulterior dificil de procesat de către alte programe, ea fiind destinată vizualizării directe în navigator.

2.1. Clase XML-RPC

Fişierele din distribuţia XML-RPC folosită sunt:

- `xmlrpc.inc` – clasele XML-RPC. Pentru a utiliza aceste clase, acest fişier trebuie inclus în programele PHP concepute.
- `xmlrpcs.inc` – clasa pentru serverul XML-RPC. Şi acest fişier trebuie inclus, alături de `xmlrpc.inc`, pentru a asigura funcţionalitatea serverului.

Cele două clase menţionate anterior sunt folosite ca biblioteci şi nu trebuie modificate. Următoarele două clase sunt utilizate pentru a crea clientul şi serverul XML-RPC, recurgând la cele două biblioteci.

- `server.php` – un server simplu cu şase funcţii: cinci pentru a decodifica *struct*-ul şi a crea comanda MySQL, şi una pentru a executa interogarea şi a întoarce rezultatul.
- `client.php` – codul clientului pentru a executa diverse funcţii în `server.php`
- `xmlrpc_client` – clasa de bază folosită pentru reprezentarea unui client al serverului XML-RPC.

Crearea unui client cu `xmlrpc_client`

Constructorul are următoarea sintaxă:

```
$client = new xmlrpc_client ($server_path,
                             $server_hostname,
                             $server_port);
```

În exemplul nostru clientul este setat pentru a interoga serverul cu adresa IP `192.168.108.60`:

```
$client = new xmlrpc_client ("RPC/server.php",
                             "192.168.108.60",80);
```

Parametrul `$server_port` este opţional; dacă este omis, ia valoarea implicită 80 când se foloseşte HTTP şi 443 în cazul utilizării protocolului HTTPS.

Metodele cele mai importante ale clasei folosite sunt:

- **send**

Această metodă se prezintă în forma:

```
$response = $client->send ($xmlrpc_message, $timeout,
                           $server_method);
```

unde `$xmlrpc_message` este o instanţă a `xmlprcmsg`, iar `$response` este instanţă a `xmlrpcresp` (obiecte XML-RPC speciale în care sunt împachetate cererea, respectiv răspunsul).

Parametrul `$timeout` este opţional şi va fi setat la 0 (aşteptare la infinit) în cazul în care este omis.

Parametrul `$server_method` este opţional; dacă este omis, ia valoarea implicită '`http`'. Singura alternativă este '`https`', care va folosi o conexiune HTTP SSL pentru conectarea la distanţă la server.

- **setDebug**

```
$client->setDebug($debugOn);
```

Parametrul `$debugOn` ia valoarea 0 sau 1 după cum se cere sau nu clientului să afişeze informaţii pentru depanare în navigatorul Web. În mod implicit aceste informaţii nu se afişează.

Informaţiile pentru depanare includ datele neprelucrate returnate de serverul XML-RPC interogat, împreună cu valoarea PHP pe care clientul încearcă să le creeze pentru a reprezenta valoarea returnată de către server. Această opţiune poate fi foarte utilă la depanarea serverelor, deoarece arată exact ce returnează serverul.

Crearea unui server cu `xmlrpc_server`

Implementarea curentă a acestei clase a fost menţinută cât mai simplă. Constructorul serverului realizează practic toată sarcina. Iată un exemplu minimal:

```
function my_select ($params) {
     // ...
}
$s = new xmlrpc_server (
            array("DB_select" =>
              array("function" => "my_select",
            "signature" => $my_select_sig,
            "docstring" => $my_select_doc)
     );
```

Aceasta realizează tot ceea ce se cere de la un server. Unicul argument este un tablou asociativ de la nume de metode la nume de funcţii. Cererea este procesată şi

expediată funcţiei relevante, care este responsabilă de returnarea unui obiect `xmlprcresp`, acesta fiind serializat şi trimis înapoi apelantului.

Primul argument al constructorului `xmlrpc_server` este un tablou numit *hartă de expediere* (*dispatch map*). În acest tablou se află informaţia de care serverul are nevoie pentru a servi metodele XML-RPC care sunt definite de utilizator.

Harta de expediere ia forma unui tablou asociativ de tablouri asociative: tabloul exterior are câte o intrare pentru fiecare metodă, cheia fiind numele metodei. Valoarea corespunzătoare este un alt tablou asociativ, care poate avea următorii membri:

- `function` – această intrare este obligatorie. Trebuie să fie un nume de funcţie aflată în domeniul global, care serveşte metoda XML-RPC;
- `signature` – este un tablou conţinând signaturile posibile pentru metodă. Dacă această intrare este prezentă atunci serverul va verifica dacă numărul şi tipul corect de parametri au fost trimişi pentru această metodă, înainte de a o expedia;
- `docstring` – un şir conţinând documentaţia pentru metodă. Documentaţia poate conţine marcaje HTML.

Signatura metodelor

O signatură este descrierea tipului returnat de metodă şi a tipurilor parametrilor ei. O metodă poate avea mai mult de o signatură.

În interiorul hărţii de expediere, fiecare metodă are un tablou de signaturi posibile. Fiecare signatură este un tablou de tipuri; prima intrare o constituie tipul întors. De exemplu, metoda:

```
my_select(struct)
```

are signatura:

```
array($xmlrpcString, $xmlrpcStruct);
```

şi, presupunând că e singura signatură posibilă pentru metodă, ea poate fi folosită pentru crearea serverului în modul următor:

```
$my_select_sig = array (array
                    ($xmlrpcString, $xmlrpcStruct));
$my_select_doc = 'Cand este apelata pasandu-i-se un struct
continand parametrii unei comenzi SQL, construieste
interogarea, interogheaza serverul BD si returneaza
rezultatul interogarii, unde tbl_id este de fapt indexul
corespunzator al tabelului interogat, intr-un tablou';
```

Pentru uşurinţă, şirurile reprezentând tipurile XML-RPC au fost codificate ca variabile globale:

```
$xmlrpcI4 = "i4";
$xmlrpcInt = "int";
$xmlrpcBoolean = "boolean";
$xmlrpcDouble = "double";
```

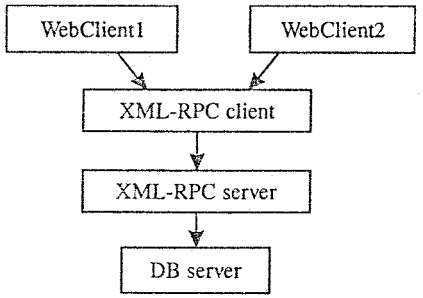
```
$xmlrpcString = "string";
$xmlrpcDateTime = "dateTime.iso8601";
$xmlrpcBase64 = "base64";
$xmlrpcArray = "array";
$xmlrpcStruct = "struct";
```

Raportarea erorilor

Pentru server, raportarea erorilor ar trebui să înceapă la valoarea indicată de variabila globală `$xmlrpcerruser+1`.

2.2. Funcţionalitatea aplicaţiei

Ideea generală de funcţionare a aplicaţiei noastre este reprezentată în schema de mai jos (după cum se observă, este vorba de un model *multi-tier*, clientul neavând direct acces la serverul de baze de date, interfaţa cu acesta realizându-se prin intermediul unor entităţi PHP care folosesc protocolul XML-RPC):



Schema arhitecturală a aplicaţiei

A. WebClientX va face o cerere către clientul XML-RPC prin intermediul unei metode numite `show_products_ofMyCompany()`, de exemplu. Clientul XML-RPC are rolul de a defini cererea către un server XML-RPC. După cum am spus mai sus, clientul Web nu cunoaşte nimic despre locaţia şi tipul bazei de date.
B. Apoi funcţia `show_products_ofMyCompany()` din clientul XML-RPC apelează o funcţie generică pentru a completa câmpurile unui obiect de tip struct, care are forma:

```
$select_param=new xmlrpcval (array(
     "field_list" => new xmlrpcval (""),
     "tbl_id"=> new xmlrpcval("6"),
     "where" => new xmlrpcval(""),
     "group by" => new xmlrpcval(""),
     "order by" => new xmlrpcval(""),
     "having" =>new xmlrpcval("")), "struct");
```


În cazul nostru `field_list` este vid, iar serverul XML-RPC va folosi „*" drept argument în comanda SQL *select*. Valoarea 6 pentru `tbl_id` este corespondentul numeric al tabelei care va fi interogată.

Următorii paşi în clientul XML-RPC sunt:

1. formulează o cerere către un server XML-RPC :

```
$f = new xmlrpcmsg('DB_select', array($select_param));
```

2. creează un şir XML care reprezintă mesajul XML-RPC:

```
htmlentities($f->serialize());
```

În cererea XML-RPC, antetul are forma:

```
POST  /RPC HTTP/1.0
User-Agent: PHP/4.0.5
Host: 192.168.108.60
Content-Type: text/xml
Content-length: 558
```

Formatul URI-ului din prima linie a antetului nu este specificat. De exemplu, vid, un caracter *slash* simplu („/"), dacă serverul tratează doar apeluri XML-RPC. Totuşi, dacă serverul tratează un amestec de cereri HTTP, se permite ca URI-ul să ruteze cererea către codul care tratează cererile XML-RPC. (În exemplu, URI-ul este chiar cuvântul `RPC`, cerând rutarea cererii către cel care răspunde la RPC.)

Trebuie specificate obligatoriu câmpurile HTTP *User-Agent* (denotă clientul Web care a formulat cererea) şi *Host* (adresa calculatorului de pe care a venit cererea din partea clientului). Câmpul *Content-Type* este `text/xml` (datele vehiculate sunt documente XML), iar *Content-Length* trebuie specificat şi să fie corect.

Apelul efectiv se află codificat în XML: o singură structură `<methodCall>`. Ea trebuie să conţină o subintrare `<methodName>`, un şir care conţine numele metodei de apelat.

Şirul poate conţine doar caractere identificator, majuscule şi minuscule, cifre, liniuţa de subliniere, punctul, punctul şi virgula, bara oblică. Modul de interpretare a caracterelor dintr-o `methodName` este în totalitate la discreţia serverului.

De exemplu, `methodName` poate fi numele unui fişier care conţine un script care se execută la apariţia unei cereri. Ar putea fi numele unei celule dintr-un tabel al unei baze de date sau calea către un fişier dintr-o ierarhie de directoare şi fişiere.

Dacă apelul de procedură are parametri, `<methodCall>` trebuie să conţină o subintrare `<params>`. Aceasta poate conţine orice număr de `<param>`, fiecare având o `<value>`.

Cererea încapsulată pentru server poate arăta astfel (în acest exemplu se observă constituenţii unei interogări SQL furnizaţi drept conţinut al elementelor XML):

```
<?xml version="1.0"?>
<methodCall>
```

```
<!-- numele metodei apelate -->
<methodName>DB_select</methodName>
   <!-- parametrii acesteia (incapsulati in struct) -->
   <params>
      <param>
         <value>
            <struct>
               <member>
                  <name>field_list</name>
                  <value><string></string></value>
               </member>
               <member>
                  <name>tbl_id</name>
                  <value><string>6</string></value>
               </member>
               <member>
                  <name>where </name>
                  <value><string></string></value>
               </member>
               <member>
                  <name>group by</name>
                  <value><string></string></value>
               </member>
               <member>
                  <name>order by</name>
                  <value><string></string></value>
               </member>
               <member>
                  <name>having</name>
                  <value><string></string></value>
               </member>
            </struct>
         </value>
      </param>
   </params>
</methodCall>
```

3. creează o instanţă a clientului:

```
$client = new xmlrpc_client ("RPC/server.php",
                             "192.168.108.60", 80);
```

4. trimite cererea încapsulată către server. Rezultatul e stocat într-o variabilă `$r`:

```
$r = $c->send($f);
```


C. Acum, funcţia `DB_select` va fi apelată în server. Acesta este numele cunoscut de client, dar serverul asociază acestui nume un nume intern diferit, `my_select`. Aici despachetăm *struct*-ul, creăm interogarea SQL, o executăm şi trimitem răspunsul clientului:

```
$sno = $m->getParam(0);
//daca exista si are tipul corect
if (isset($sno) && ($sno->scalartyp() == "struct")) {
 //creeaza interogarea SQL particulara serverului SQL
 //(aici MySQL)
 //apeleaza o functie care va executa efectiv comanda SQL:
 query_result($query, $func_name)
 //returneaza clientului XML-RPC rezultatul interogarii:
 new xmlrpcresp($q_result);
 //unde $q_result este recordset-ul incapsulat XML
 //aceasta se face cu ajutorul unei functii
 rst2xml($recordSet_object);
}
```

D. Iată formatul antetului răspunsului în exemplul nostru:

```
HTTP/1.1 200 OK
Date: Mon, 28 Apr 2003 05:28:53 GMT
Server: Apache/1.3.14 (Win32)
X-Powered-By: PHP/4.0.5
Content-length: 148
Connection: close
Content-Type: text/xml
```

Cu excepţia cazului în care există o eroare la un nivel inferior, se returnează întotdeauna codul HTTP de succes `200 OK`.

Şi în acest caz, *Content-Type* este `text/xml` (se întoarce rezultatul codificat în XML), iar *Content-Length* trebuie să fie prezent şi corect.

Corpul răspunsului este o singură structură XML, un element `<methodResponse>` care poate conţine un singur element `<params>` având ca element-copil un `<record>` care include câmpurile unui *recordset* şi valorile asociate acestora.

La rândul ei, `<methodResponse>` poate conţine şi un element `<fault>`; acesta include un `<value>`, care e un `<struct>` având două elemente, unul numit `<faultCode>` (un element `<int>`) şi unul numit `<faultString>` (un `<string>`) – astfel se poate returna clientului un mesaj de eroare în cazul în care apelul serviciului dorit a eşuat.

Un element `<methodResponse>` nu poate conţine simultan elementele `<fault>` şi `<params>`.

Clientul XML-RPC va primi un răspuns care arată astfel (datele returnate în urma execuţiei interogării SQL – în acest exemplu presupunem că s-au returnat informaţii privitoare la componente de tip memorie RAM):

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params>
<record1>
  <!-- identificatorul unic al produsului -->
  <pr_id>
     <value>
       <int>
         74
       <int>
     </value>
  <pr_id>
  <!-- codul produsului -->
  <prod_code>
     <value>
       <string>
         0031
       <string>
     </value>
  </prod_code>
  <!-- numele produsului -->
  <prod_name>
     <value>
       <string>
         Memorie DDR
       <string>
     </value>
  </prod_name>
  <!-- unitatea de masura -->
  <prod_measure_unit>
     <value>
       <string>
         MB
       <string>
     </value>
  </prod_measure_unit>
  <!-- pretul -->
  <prod_price>
     <value>
       <double>
         105.5
       <double>
     </value>
  </prod_price>
```



```
</record1>
<record2>
    <!-- alte inregistrari returnate... -->
</record2>
</params>
</methodResponse>
```

2.3. Biblioteca XML-RPC pentru PHP furnizată de *Incutio*

Pentru un nivel superior de control al erorilor şi pentru o mai mare simplitate în implementarea clientului şi a serverului XML-RPC, se pot folosi clasele *Incutio*, iar pentru optimizarea lucrului cu bazele de date tehnologia *ADOdb*. Formatul de returnare al răspunsurilor de la server este acum un tablou. Funcţionalitatea de bază rămâne aceeaşi, ambele variante putând fi utilizate pentru atingerea scopului propus.

Biblioteca distribuţiei Incutio conţine un singur fişier, `IXR_Library.inc.php`. Acesta este fişierul bibliotecă care înglobează clasele XML-RPC atât pentru client cât şi pentru server (`IXR_Client`, `IXR_Server` şi `IXR_IntrospectionServer`). De asemenea, mai există clasele `IXR_Error`, `IXR_Message`, `IXR_Value`, `IXR_Request`, `IXR_Date`, `IXR_Base64`, `IXR_ClientMulticall`.

Ca şi în distribuţia anterior menţionată (a se vedea secţiunea 2.1 a prezentului capitol), fişierul este folosit ca bibliotecă în crearea clientului şi a serverului XML-RPC şi nu trebuie modificat.

Cererea

În IXR (*Incutio XML-RPC*), o cerere XML-RPC are forma:

```
$client->query ('nume.metoda', $parametrul1, $parametrul2,
                ...);
```

Parametrii pot fi variabile PHP obişnuite, tablouri PHP simple şi indexate şi chiar obiecte PHP (care sunt convertite la *struct*-uri de către bibliotecă). Se observă că parametrii sunt transmişi mult mai uşor, nefiind nevoie să fie încapsulaţi într-un obiect `xmlprcmsg` ca în metoda anterior descrisă.

Răspunsul

Există posibilitatea ca în urma fiecărui apel de procedură să se returneze o cantitate nelimitată de date structurate. IXR asociază cererile XML-RPC direct unor funcţii PHP sau către metodele claselor şi realizează automat conversia tipurilor – tot ceea ce vă rămâne de făcut este scrierea unei funcţii care returnează o valoare (sau un tablou), iar IXR va avea grijă de rest.

Erorile

În XML-RPC, erorile sunt reprezentate printr-un cod de eroare numeric şi un mesaj de eroare descris de un şir. IXR reprezintă aceste erori ca obiecte `IXR_Error`

Biblioteca va genera automat aceste obiecte ori de câte ori ceva a funcţionat defectuos, iar aceste obiecte vă vor sta la dispoziţie pentru procesare de către propriul dumneavoastră cod de tratare a erorilor.

Specificaţia pentru interoperabilitatea codului de eroare

IXR suportă specificaţia pentru interoperabilitatea codului de eroare – orice eroare din bibliotecă care se potriveşte cu unul dintre codurile de eroare definite în specificaţie este returnată cu codul de eroare corespunzător.

Conversia automată a tipurilor

IXR se străduieşte să ascundă utilizatorului cât mai mult din detaliile specificaţiei XML-RPC. Ori de câte ori este posibil, tipurile XML-RPC sunt convertite în tipuri PHP şi invers, fără a fi nevoie de intervenţia programatorului serviciului Web. Când acest lucru este imposibil, sunt returnate obiecte care permit accesul uşor la datele încapsulate.

Extensii XML-RPC suportate

- **system.multicall**

Specificaţia *system.multicall* este implementată complet în clasa `IXR_Server`. Toate serverele construite cu această clasă (sau cu clasa `IXR_IntrospectionServer`, care o extinde) vor avea disponibilă metoda *system.multicall*, permiţând clienţilor să efectueze cereri de proceduri multiple într-o tranzacţie HTTP.

Mai multe detalii pot fi găsite la următoarea adresă Web: **http://www.xmlrpc.com/discuss/msgReader$1208**.

Suportul client pentru *system.multicall* este disponibil în clasa `IXR_Client Multicall`, care extinde `IXR_Client`.

- **system.getCapabilities**

Metoda `system.getCapabilities` este implementată în clasa de bază `IXR_Server` şi este actualizată automat de către clasa `IXR_IntrospectionServer` pentru a asigura faptul că returnează informaţii despre capacităţile suplimentare provenite din utilizarea clasei pentru un server mai complex.

- **Introspecţia**

IXR_Server pune la dispoziţie metoda `system.listMethods()` pentru toate serverele XML-RPC construite folosind biblioteca. `system.methodSignature()` şi `system.methodHelp()` nu sunt suportate în clasa server de bază, dar pot fi complet implementate folosind `IXR_IntrospectionServer`. Ambele metode impun ca programatorul serviciului Web să furnizeze informaţii suplimentare despre metodele pe care le defineşte – din acest motiv ele au fost omise din `IXR_Server` pentru a menţine implementarea de bază cât mai simplă posibil.

Construcţia clientului de bază

Biblioteca IXR permite efectuarea cererilor către serverele XML-RPC folosind clasa `IXR_Client`. Înainte de crearea unei conexiuni este nevoie de cunoaşterea calculatorului *host*, a căii şi a portului serverului XML-RPC cu care se doreşte comunicarea, precum şi a metodei ce va fi apelată pe server.

Mai întâi este necesară crearea unui obiect `IXR_Client`. Trebuie furnizate detalii legate de severul la care se doreşte conectarea atunci când are loc crearea unui obiect, ca URL sau sub forma nume *host*, cale şi port:

```
// Specificarea/Identificarea serverului prin URL
// (implicit - port 80):
$client = new IXR_Client
            ('http://myAddress.com/xmlrpc/myserver.php');
// Identificarea serverului prin IP-ul
// calculatorului host, cale si port
$client = new IXR_Client(' myAddress.com ',
                         '/xmlrpc/myserver.php', 80);
```

Pasul următor este efectuarea apelului XML-RPC folosind metoda `query()`. Metoda acceptă un *numeDeMetodă* XML-RPC cu un număr variabil de argumente care vor fi pasate serverului împreună cu cererea. Metoda `query()` returnează *true* sau *false* în funcţie de succesul sau eşecul cererii. În cazul în care cererea eşuează, metodele `getErrorCode()` şi `getErrorMessage()` furnizează o explicaţie detaliată a erorii:

```
if (!$client->query('demo.squareVal', 3)) {
    die('Eroare de executie - ' .
          $client->getErrorCode() . ' : ' .
          $client->getErrorMessage());
}
```

Presupunând că metoda s-a executat cu succes, datele returnate ca răspuns pot fi accesate folosind metoda `getResponse()`, având forma unei valori sau a unui tablou de valori, în funcţie de structura de date returnată de server.

```
echo $client->getResponse();
```

Depanare

Dacă este necesară depanarea apelurilor de procedură client, aceasta se poate realiza prin setarea *flag*-ului de depanare.

```
$client = new IXR_Client
            ('http://myAddress.com/xmlrpc/myserver.php');
$client->debug = true;
// Rulam un script PHP
```

```
if (!$client->query(parameters)) {
    die(' Eroare de executie - ' .
    $client->getErrorCode() . ' : ' .
    $client->getErrorMessage());
}
```

Cu *flag*-ul depanare setat, `IXR_Client` va afişa complet toate mesajele XML-RPC emise şi recepţionate.

Construcţia unui server de bază

Serverele de bază în IXR sunt construite folosind clasa `IXR_Server`. Metodologia este simplă: se creează funcţii PHP (sau metode de clase dacă se merge pe drumul programării obiectuale), apoi se apelează constructorul `IXR_Server` având ca argument un tablou care asociază (*mapping*) nume de metode XML-RPC unor nume ale funcţiilor proprii. Clasa `IXR_Server` va apela apoi funcţiile relevante pe baza informaţiilor provenite de la client.

Funcţiile apelate pot returna variabile PHP obişnuite (şiruri, întregi, tablouri sau chiar obiecte), iar IXR va converti valorile returnate într-un răspuns XML-RPC şi îl va trimite clientului.

Iată un server IXR simplu:

```
<?php
include('IXR_Library.inc.php');

// implementeaza o functie de aflare a datei
function getDate($args) {
    return date('d-M-Y');
}
// implementeaza o functie de ridicare la patrat
function squareVal($args) {
    return $args[0] * $args[0];
}
// initializare
$server = new IXR_Server(array(
    'demo.getDate' => 'getDate',
    'demo.squareVal' => 'squareVal'
));
?>
```

Codul de mai sus va crea un server XML-RPC dispunând de două metode: `demo.getDate()` care returnează data calendaristică în formatul zi-lună-an şi `demo.squareVal()` care acceptă o valoare numerică şi returnează pătratul acesteia.

Se observă că ambele funcţii acceptă un singur argument: `$args`; acesta este un tablou de parametri primiţi de la clientul XML-RPC. Dacă există un singur argument, `$args` va fi o variabilă de tip simplu şi nu un tablou având un element unic.

Transmiterea erorilor

Mecanismul XML-RPC posedă un sistem de erori flexibil care poate fi utilizat în implementările serverului. O eroare XML-RPC constă dintr-un număr de eroare (un întreg) şi un şir descriptiv. Numărul utilizat depinde de dezvoltatorul aplicaţiei, dar este important ca schema de numere considerată să fie consistentă. Erorile pot fi trimise clientului folosind următorul cod:

```
function myCallbackFunction($args) {
    if ($args[0] != 'parolaSecreta') {
        return new IXR_Error (-1,
                'Nu ati introdus parola corecta!');
    }
    //Restul functiei
}
```

Funcţia de mai sus va returna o eroare dacă primul argument trimis de client este orice altceva decât '`parolaSecreta`' – poate fi folosit pentru a forma un sistem de autentificare elementar (nu neapărat sigur).

Date calendaristice şi date în codificarea base64

XML-RPC oferă suport pentru două tipuri care nu au corespondent în PHP – date calendaristice şi date în codificarea *base64* (a se revedea şi cele descrise în secţiunea 3.3 a capitolului al doilea). Pentru a returna datele în aceste formate, se foloseşte:

```
function using_b64($args) {
    $data = 'sir care va fi transmis codificat in base64';
    $time = time();
    return array(
        'data' => new IXR_Base64($data),
        'time' => new IXR_Date($time)
    );
}
```

Funcţia de mai sus returnează un tablou indexat (un *struct* XML-RPC) constând într-o porţiune de date codificate *base64* şi o dată calendaristică codificată ISO. După cum se poate vedea, detaliile codificării sunt ţinute ascunse faţă de programatorul serviciului Web.

Abordarea orientată-obiect

Alături de crearea de servicii Web cu funcţii *callback*, IXR poate fi utilizat şi în dezvoltarea de servicii Web prin clase PHP. Acest lucru poate fi obţinut prin crearea unei clase server care extinde `IXR_Server` şi definirea metodelor care să fie folosite ca proceduri XML-RPC în noua clasă. În acest moment poate fi apelat constructorul clasei `IXR_Server`, cu deja obişnuitul tablou de *callback*, cu o singură mică diferenţă: fiecare dintre funcţiile de *callback* trebuie prefixată cu „`this:`", pentru a aduce la cunoştinţa IXR că se apelează de fapt o metodă de clasă şi nu o funcţie obişnuită.

Iată serverul de mai sus implementat utilizând paradigma obiectuală:

```
<?php
include('IXR_Library.inc.php');
class SimpleServer extends IXR_Server {
    // constructor
    function SimpleServer {
        $this->IXR_Server(array(
            'demo.getDate' => 'this:getDate',
            'demo.squareVal' => 'this:squareVal'
        ));
    }

    // metode implementate
    function getDate($args) {
        return date('d M Y');
    }
    function squareVal($args) {
        return $args[0]*$args[0];
    }
}
// instantierea unui serviciu Web folosind IXR
$server = new SimpleServer();
?>
```

2.4. Biblioteca PHP *ADOdb* pentru lucrul cu bazele de date

Funcţiile PHP pentru acces la bazele de date nu sunt standardizate. Acest lucru conduce la nevoia unei biblioteci de clase pentru baze de date care să ascundă diferenţele între API-urile diverselor servere de baze de date (să încapsuleze particularităţile implementărilor acestora) astfel încât interschimbul de informaţii privitoare la bazele de date şi conţinutul lor să se realizeze facil. O implementare care satisface aceste cerinţe este *ADOdb* (*Active Data Objects Data Base* – a se vedea **http://php.weblogs.com/adodb**), necesitând o versiune PHP 4.0.5 sau una superioară deoarece se foloseşte funcţia `str_replace()` pe tablouri. Pentru detalii de instalare se poate vizita adresa **http://phplens.com/lens/adodb/docs-adodb.htm#install**.

În momentul de faţă biblioteca *ADOdb* suportă MySQL, Oracle, Microsoft SQL Server, Sybase, Sybase SQL Anywhere, Informix, PostgreSQL, FrontBase, Interbase (variantele Borland şi Firebird), FoxPro, Access, ADO and ODBC; de asemenea, s-a reuşit conectarea cu succes la Progress şi DB2 prin ODBC. Putem privi *ADOdb* ca fiind similara modulelor *DBI* de manipulare a bazelor de date în Perl.

Cum PHP4 suportă variabile de sesiune, acestea pot fi stocate folosind biblioteca ADOdb pentru o portabilitate şi o scalabilitate reală.

Printre caracteristicile specifice ADOdb se pot enumera:

- Uşurinţă în adaptare a programatorilor Windows, deoarece multe dintre convenţii sunt similare cu cele oferite de *ADO* (*ActiveX Data Objects*) de la Microsoft – a se consulta, de exemplu, cartea *Programming ADO* de D. Sceppa, Microsoft Press, 2000.
- Spre deosebire de alte clase PHP pentru baze de date, care se concentrează doar pe comenzi SQL *select*, este furnizat cod-suport pentru a trata *insert*-uri şi *update*-uri care pot fi repede adaptate la mai multe baze de date. Sunt puse la dispoziţie metode pentru manipularea datelor, concatenarea şirurilor şi caractere de demarcare a şirurilor pentru diferite sisteme de baze de date.
- Un sistem de meta-tipuri este construit astfel încât se poate determina când tipuri precum CHAR, TEXT şi STRING sunt echivalente în diferite baze de date.
- Uşurinţă în portare datorită faptului că tot codul dependent de baza de date este stocat în funcţii *stub*. Nu este necesară portarea logicii principale a claselor;
- Suport pentru sesiuni PHP4.

O serie de aplicaţii care utilizează ADOdb sunt:

- *PhpLens* este o componentă comercială tip grilă de date care permite atât designerilor Web, cât şi altor programatori să dezvolte şi să menţină uşor baze de date pe Web; este dezvoltată chiar de autorul bibliotecii ADOdb.
- *PHAkt: PHP Extension for Dreamweaver Ultradev* permite introducerea scripturilor PHP în popularul editor de pagini Web Macromedia Dreamweaver. Manipularea bazelor de date se face cu ADOdb.
- *PostNuke* este un sistem gratuit foarte popular de gestiune a conţinutului şi de monitorizare a traficului Web. Oferă suport complet CSS, concordanţă cu HTML 4.01 tranziţional şi are suport multilingvistic complet.
- *EasyPublish CMS* este un alt sistem gratuit de gestiune a conţinutului pentru informaţii şi module integrate în situri Internet, intranet şi extranet.

Un exemplu – comanda Select

Pentru exemplificarea modului de lucru cu bazele de date al acestei biblioteci, prezentăm un exemplu dat de autorii acesteia: conectarea la o bază de date Microsoft Access (de exemplu, *Northwind*) şi afişarea primelor două coloane din fiecare rând.

Mai întâi se creează un obiect `ADOConnection` care reprezintă conexiunea la baza de date. Această conexiune este iniţiată cu `PConnect`, care este o conexiune persistentă. De fiecare dată când se doreşte interogarea bazei de date este apelată metoda `ADOConnection.Execute()`. Aceasta returnează un obiect `ADORecordSet` care este în fapt un cursor păstrând înregistrarea curentă în tabloul `fields[]`. Pentru deplasarea de-a lungul înregistrărilor returnate se foloseşte `MoveNext()`. De asemenea, o funcţie utilă – nefolosită însă în aplicaţia noastră – este `SelectLimit()`, permiţând limitarea numărului de înregistrări afişate.

```
<?
include('adodb.inc.php');
// incarcarea codului general ADOdb
```

```
$conn = &ADONewConnection('access'); // crearea conexiunii
$conn->PConnect('northwind');
// conectarea la Access,northwind DSN
$recordSet = &$conn->Execute('select * from products');
if (!$recordSet)
    print $conn->ErrorMsg();
else
while (!$recordSet->EOF) {
    print $recordSet->fields[0].' '.
    $recordSet->fields[1].'<BR>';
    $recordSet->MoveNext();
}
$recordSet->Close();                    // optional
$conn->Close();                         // optional
?>
```

Variabila `$recordSet` returnată stochează înregistrarea curentă din tabloul `$recordSet->fields`, indexat după numărul coloanei (începând cu zero). Proprietatea EOF este setată la *true* când este găsit sfârşitul de fişier. Dacă apare o eroare în `Execute()`, se returnează *false* în locul unui *recordset*.

Tabloul `$recordSet->fields[]` este generat de extensia PHP pentru baze de date. Unele extensii pentru baze de date indexează tabloul doar după număr şi nu după numele câmpului. Pentru a forţa indexarea după nume – adică transformarea tabloului într-unul asociativ – se foloseşte funcţia `SetFetchMode()`. Fiecare *recordset* salvează şi foloseşte modul de returnare a înregistrării care a fost stabilit la crearea *recordset*-ului cu `Execute()` sau `SelectLimit()`.

```
$db->SetFetchMode(ADODB_FETCH_NUM);
$rs1 = $db->Execute('select * from table');
$db->SetFetchMode(ADODB_FETCH_ASSOC);
$rs2 = $db->Execute('select * from table');
print_r($rs1->fields);
  # afiseaza array([0]=>'v0',[1] =>'v1')
print_r($rs2->fields);
  # afiseaza array(['col1']=>'v0',['col2'] =>'v1')
```

Pentru a obţine numărul de înregistrări dintr-o comandă *select*, se poate folosi `$recordSet->RecordCount()`, care poate returna -1 dacă numărul de înregistrări nu poate fi determinat.

Obiectul principal cu care se realizează interogările în baza de date este `ADOConnection`. Prin intermediul acestuia se realizează conectarea la baza de date şi apoi se execută comenzile SQL; conţine un set de funcţii utile pentru standardizarea formatului comenzilor SQL pentru probleme precum concatenarea şirurilor şi formatul datelor calendaristice.

Vom prezenta o singură metodă a unui astfel de obiect, restul funcţionalităţilor putând fi descoperite şi studiate în manualul de utilizare a bibliotecii. Metoda `Execute($sql, $inputarr=false)` execută comanda SQL stocată în variabila `$sql` şi în caz de succes returnează clasa derivată din `ADORecordSet`. Trebuie observat faptul că în caz de succes ea returnează un *recordset*, chiar dacă se execută o comandă *insert* sau *update*. De exemplu, în conexiunea cu un server MySQL va fi returnat un `ADORecordSet_mysql`. Dacă apare o eroare de execuţie, se returnează *false*.

Parametrul `$inputarr` poate fi folosit pentru legarea variabilelor de parametri; iată un exemplu Oracle:

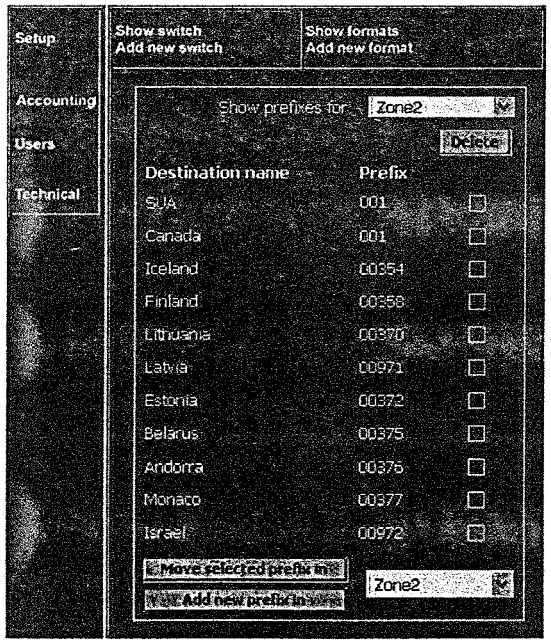
```
$conn->Execute("SELECT * FROM TABLE WHERE COND=:val",
                array('val'=> $val));
```

3. Aplicaţia *PhoneBillings*

După cum deja am descris în cadrul primei abordări XML-RPC (vezi subcapitolul 2), este vorba de interfaţa Web cu o bază de date furnizând informaţii pentru un modul de calcul al facturii telefonice.

Aplicaţia se bazează pe protocolul XML-RPC – abordarea *Incutio* alături de tehnologia ADOdb; am folosit PHP 4.2.3, MySQL 3.23.56 şi Apache 2.0.

Pentru că deja am tratat subiectul la nivel teoretic, în continuare vom vedea cum sunt folosite tehnicile de mai sus la implementarea propriu-zisă. Astfel, în figura următoare, urmând legătura *Show prefixes* clientul apelează pe rând o funcţie care completează lista cu grupuri, apoi una care returnează prefixele din grupul selectat în lista anterioară:



Modulul de gestionare a prefixelor

Pentru completarea primei liste – cea incluzând toate grupurile luate în calculul facturii telefonice de către o companie de telefonie fixă, de exemplu –, în clientul Web are loc un apel de funcţie:

```
get_groups(-5);
```

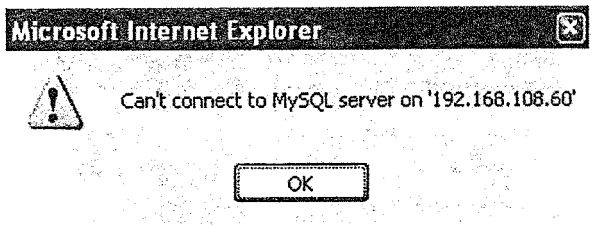
Aceasta este o funcţie apelată în client care va returna toate grupurile de destinaţii ale convorbirilor, mai puţin pe cel care are identificatorul numeric unic (ID) dat ca parametru funcţiei. În cazul nostru vor fi returnate toate grupurile existente, datorită faptului că toate au un identificator pozitiv.

În client, funcţia are forma:

```
function get_groups($excluded_group) {
  /* se creeaza tabloul care va contine
     rezultatul interogarii */
  $groups_list = array();
  /* se incearca conectarea la baza de date; in caz de
    succes se returneaza un obiect ADOConnection de lucru
    cu baza de date */
  $connObj = get_connected();
  /* se verifica intai daca au fost probleme de conexiune
     la serverul de baze de date; in acest caz semnalam
     eroarea, fara a executa propriu-zis interogarea: */
  for ($i = 0; $i < count($connObj->lastQueryErr); $i++) {
     if (array_key_exists ('faultCode',
                        $connObj->lastQueryErr[$i])) {
       $groups_list[0]=
          $connObj->lastQueryErr[$i]['faultCode'] . " " .
          $connObj->lastQueryErr[$i]['faultString'];
     return $groups_list;
     }
   }
```

Erorile sunt stocate într-un tablou de tablouri, `lastQueryErr`.

Astfel se trimite deja răspunsul scriptului PHP care va încărca un şablon (*template*) HTML, generând un cod JavaScript de forma:



Altfel putem începe execuţia comenzii SQL. Construim mai întâi interogarea SQL:


```
$sql_exec = 'select concat(coalesce(group_ID,""),\':\',
             coalesce(group_name,""))
             from tbl_dest_groups
             where group_ID<>' . $excluded_group .
            ' order by group_ID';
$res = $connObj->Execute($sql_exec);
```

Câteva detalii ale funcţiei `Execute()` din client:

```
function Execute($sql, $inputarr=false) {
    // ...
    $this->multicallClient->addCall
       ('adodbrpc.callDirect', 'Execute',
         array($sql, $inputarr));
}
```

Se observă apelul *multicall* şi numele funcţiei apelate la server `adodbrpc.callDirect()`.

Ca mai sus, verificăm dacă operaţia a avut loc cu succes:

```
  if (!$res) {
   for($i = 0;$i < count($connObj->lastQueryErr); $i++) {
    if(array_key_exists('faultCode',
                        $connObj->lastQueryErr[$i])){
     $groups_list[0]=
         $connObj->lastQueryErr[$i]['faultCode']." ".
         $connObj->lastQueryErr[$i]['faultString'];
     return $groups_list;
    }
   }
  }
```

În caz contrar, preluăm rezultatul:

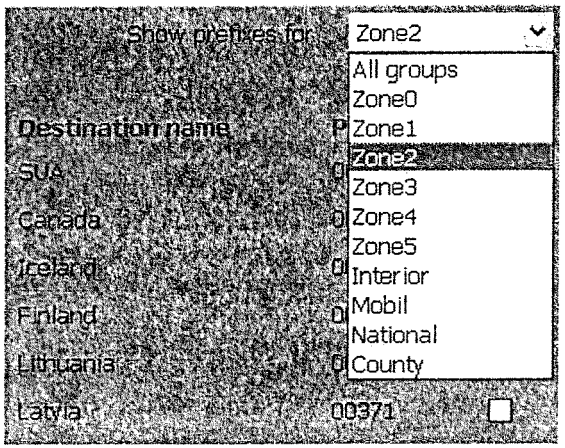
```
$connObj->getResponse();
```

Răspunsul concret are forma:

```
Array(
  [0] => Array
   (
    [group_info] => 0:Zone0
   )
  [1] => Array
   (
    [group_info] => 1:Zone1
   )
  [2] => Array
```

```
    (
     [group_info] => 2:Zone2
    )
   [3] => Array
    (
     [group_info] => 3:Zone3
    )
    // restul inregistrarilor obtinute in urma interogarii
)
```

Acest şir este trimis scriptului PHP care va încărca şablonul de afişare pentru ca utilizatorul să obţină lista:



Lista grupurilor de destinaţii posibile

Revenim puţin asupra funcţiei care face conectarea la serverul de baze de date:

```
function get_connected() {
   $ado_connection = new adodbrpc
('http://192.168.108.60/server/adodb-xmlrpc.srv.inc.php');
   $driver = 'mysql';
   $res = $ado_connection->ADONewConnection($driver);
   $ado_connection->debug = false;
   $ado_connection->ADODB_FETCH_MODE = ADODB_FETCH_ASSOC;
   $ado_connection->retFormat = 'array';
   if (!$res){
     return $ado_connection;
   }
   $res = $ado_connection->Connect ('192.168.108.60',
            'user', 'password','phoneBillings');
```



```
     return $ado_connection;
  }
```

Metoda `adodbrpc()` este constructorul clasei cu acelaşi nume, clasa care construieşte efectiv clientul XML-RPC, înglobând şi funcţionalitatea bibliotecii PHP ADOdb:

```
function adodbrpc($host, $path=false, $port = 80) {
      $this->client = new IXR_Client($host, $path, $port);
  }
```

Se observă că erorile sunt capturate la fiecare nivel. Aruncăm o scurtă privire asupra cererii clientului şi al răspunsului primit de la server:

```
POST /server/adodb-xmlrpc.srv.inc.php HTTP/1.0
Host: 192.168.108.60
Content-Type: text/xml
User-Agent: The Incutio XML-RPC PHP Library (multicall client)
Content-length: 1726
<?xml version="1.0"?>
<methodCall>
<methodName>system.multicall</methodName>
<params>
  <param><value><array><data>
    <value>
    <struct>
      <member><name>methodName</name><value><string>
       adodbrpc.ADONewConnection</string></value></member>
      <member><name>params</name>
        <value><array><data>
          <value><string>mysql</string></value>
        </data></array></value>
      </member>
    </struct>
    </value>
. . .
<struct>
  <member><name>methodName</name><value><string>
    adodb rpc.Connect</string></value></member>
  <member><name>params</name>
    <value><array><data>
      <value><string>192.168.108.60</string></value>
      <value><array><data>
        <value><string>my_login</string></value>
        <value><string>my_password</string></value>
        <value><string>phoneBillings</string></value>
      </data></array></value>
```

```
    </data></array></value>
  </member>
</struct>
</value>
</data></array></value></param>
</params>
</methodCall>
```

Prin apelul `$ado_connection->Connect()` din clientul XML-RPC sunt trimişi toţi parametrii necesari dialogului cu baza de date. Astfel prin specificaţia `system.multicall` alături de apelurile `adodbrpc.ADONewConnection()` şi `adodbrpc.Connect()` au loc şi `adodbrpc.ADOConnectionFieldSet()`, `adodbrpc.outputFormat()`, `adodbrpc.ADODB_FETCH_MODE`.

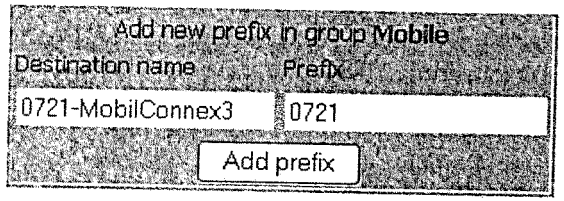
```
<?xml version="1.0"?>
<methodResponse>
<params>
 <param>
  <value>
    <array><data>
    <value>
        <array><data>
           <value><boolean>1</boolean></value>
        </data></array>
    </value>
    <!-- si alte informatii... -->
    <value>
     <struct>
      <member><name>faultCode</name>
        <value><int>4000</int></value>
      </member>
      <member><name>faultString</name>
       <value>
        <string>mysql error: [2003:
            Can't connect to MySQL server on
            '192.168.108.60' (10061)] in
            CONNECT('192.168.108.60', ?, ?,
            'phoneBillings')</string>
       </value>
      </member>
     </struct>
    </value>
   </data></array>
  </value>
 </param>
```



```
</params>
</methodResponse>
```

În răspuns se poate determina care dintre operaţii s-a încheiat cu succes. Astfel, cererea `adodbrpc.ADONewConnection()` cu parametrul „`mysql`" returnează *true*. La cererea `adodbrpc.Connect()` – cu parametrii adresa IP a serverului de baze de date, numele de cont (*login*), parola, numele bazei de date –, răspunsul furnizat este o eroare de execuţie.

Revenind la interfaţa Web, la adăugarea unui nou prefix, introducem o valoare deja existentă în baza de date (în figura care prezintă *Modulul de gestionare al prefixelor* alegem în lista de jos grupul în care vom adăuga prefixul, apăsând apoi **Add new prefix**):



Adăugarea unui prefix

În scriptul PHP se apelează funcţia `add_prefix($group_ID, $prefix, $description)` din clientul XML-RPC, funcţie care urmează aceiaşi paşi ca în cazul unui *select*. Singura deosebire este că, în caz de succes, se returnează un tablou vid. Revenind la cazul nostru, va fi semnalată eroarea de mai jos din care se va extrage mesajul de eroare prezent în căsuţa de dialog JavaScript:

```
Array
(
    [0] => 4000
    [1] => mysql error: [1062: Duplicate entry
'0721-0721-MobilConnex3' for key 1] in EXECUTE("insert into
dest_prefixes values( "0721","0721-MobilConnex3")")
(1062)
)
```

Serverul XML-RPC este construit prin extinderea clasei `IXR_IntrospectionServer`. Se observă mai jos modul în care se realizează asociarea (*mapping*) funcţiilor apelate pe server :

```
class xmlrpcAdodbServer extends IXR_IntrospectionServer {
// alte definitii...
  function xmlrpcAdodbServer() {
        $this->IXR_IntrospectionServer();
        . . .
        $this->addCallback('adodbrpc.ADODB_FETCH_MODE',
```

```
        'this:ADODB_FETCH_MODE', array('int', 'int'),
        'Set/Get the global $ADODB_FETCH_MODE!');

        $this->addCallback('adodbrpc.callDirect',
  'this:callDirect', array('struct', 'string', 'array'),
  'Run adodb natives methods and return the result!');
        $this->addCallback('adodbrpc.Connect',
  'this:Connect', array('boolean', 'string', 'array'),
  'connection, and return true if success');
  }
// ... (metodele serverului)
}

  function addCallback($method, $callback, $args, $help) {
        $this->callbacks[$method] = $callback;
        $this->signatures[$method] = $args;
        $this->help[$method] = $help;
  } //... in IXR_Library
```

Metoda `addCallback()` creează corespondenţa între funcţiile apelate de client în mod formal şi cele apelate efectiv pe server. Ea primeşte patru parametri, respectiv numele metodei XML-RPC (apelate de client), numele funcţiei de executat pe server, un tablou specificând signatura funcţiei de apelat şi un şir care documentează metoda XML-RPC.

Datorită complexităţii acestei clase, precum şi datorită faptului că aplicaţia este în curs de dezvoltare, ne vom limita la explicaţiile de mai sus în ceea ce priveşte implementarea serverului.

4. Concluzii

Am prezentat principalele trăsături ale protocolului XML-RPC, prin care este posibilă apelarea de proceduri la distanţă – baza pentru implementarea de servicii Web –, mesajele schimbate fiind stocate în format XML. De asemenea, am exemplificat folosirea tehnologiei XML-RPC prin gestiunea unei baze de date direct pe Internet prin intermediul unei aplicaţii Web de calcul a facturilor telefonice. Îndemnăm cititorul să încerce să implementeze diverse aplicaţii (inclusiv cele descrise în capitolele anterioare ale cărţii de faţă) prin adoptarea tehnologiilor de prelucrare a documentelor XML şi a bazelor de date prezentate în acest capitol.

Avantajele aduse de XML-RPC fac din această tehnologie o opţiune demnă de luat în seamă în proiectarea aplicaţiilor Web distribuite, concepute exclusiv cu tehnologii deschise.

Comentarii bibliografice

Pentru alte detalii privind utilizarea paradigmei orientate-obiect în cadrul programelor PHP a se vedea volumul II, partea G din S. Buraga, *Tehnologii Web*, Matrix Rom, Bucureşti, 2001.

Mai multe despre SOAP şi servicii Web şi în articolul „Dialoguri despre SOAP" de L. Alboaie şi S. Buraga, *NET Report*, feb. 2003 (disponibil pe Web la **http://www.infoiasi.ro/~busaco/publications/articles/SOAP.pdf**). De asemenea, recomandăm vizitarea sitului Consorţiului Web, secţiunea dedicată activităţii de dezvoltare a serviciilor Web: **www.w3.org/2001/ws/**.

Cititorii interesaţi pot consulta şi următoarele adrese Web:

- Specificaţia XML-RPC: **http://www.xmlrpc.com/spec**
- D. Winer, *XML-RPC for Newbies*: **http://davenet.userland.com/1998/07/14/xmlRpcForNewbies**
- Un tutorial XML-RPC: **http://www.wsjug.org/archives/11_06_01/**
- Specificaţia SOAP: **http://www.w3.org/TR/SOAP/**
- K.Rhodes, *XML-RPC vs. SOAP*: **http://weblog.masukomi.org/writings/xml-rpc_vs_soap.htm**
- Situl Web pentru clasele *Incutio*: **http://scripts.incutio.com**
- Situl Web al bibliotecii *ADOdb*: **http://php.weblogs.com/adodb**
- Lumea serviciilor Web: **http://www.webservices.org**
- Situl Web al aplicaţiei *VisualOffice*: **http://www.mintercorp.com**
- Situl Web al organizaţiei *WebCrossing*: **http://www.webcrossing.com**

CAPITOLUL 8

Aplicaţii propuse

Sabin Buraga, busaco@infoiasi.ro

Capitolul de faţă propune o serie de aplicaţii Web care să fie implementate de către cititori, folosind cunoştinţele acumulate în precedentele capitole.

Lista aplicaţiilor pe care cititorul este invitat să le dezvolte (eventual, doar prin prisma anumitor aspecte, precum proiectarea interfeţei Web, structurarea informaţiilor în baze de date sau prin intermediul documentelor XML ori implementarea unor module semnificative) este următoarea:

- **WebIdent**

Pentru autentificarea uniformă a utilizatorilor, în cadrul diferitelor aplicaţii Web cu caracter educaţional (vezi mai jos), trebuie conceput un modul de identificare unică pe baza numelor de utilizator şi parole. Modulul va trebui să furnizeze, în caz de autentificare soldată cu succes, tipul de utilizator autentificat (student, cadru didactic, membru al secretariatului etc.), plus alte informaţii utile (de exemplu, pentru un student ar trebui returnate: anul din care provine, numărul matricol, numele real etc.). Acest modul ar putea funcţiona şi ca serviciu Web, informaţiile returnate fiind marcate într-un limbaj bazat pe XML.

- **WebRights**

În conjuncţie cu modulul *WebIdent*, să se implementeze un sistem de stabilire a drepturilor de acces la resursele din intranetul unei organizaţii (academice). Acest sistem trebuie să includă diferite niveluri de acces, de genul:

- acces individual – pe bază de autentificare cu nume şi parolă (de exemplu, accesul unui cadru didactic pentru modificarea temelor de licenţă propuse pentru anul viitor);
- acces al unui grup – pe bază de autentificare la nivelul grupului, de exemplu prin verificarea apartenenţei la grup(uri); *i.e.* accesul studenţilor pasionaţi de tehnologii Web la paginile dedicate acestui subiect sau accesul membrilor consiliului facultăţii la diverse informaţii privitoare la activitatea acestuia;
- acces în domeniul intranetului organizaţiei (de exemplu doar din domeniul organizatia.ro), pe baza adresei IP a navigatorului (de exemplu, accesul la resursele bibliografice sau la listele de discuţii interne).

De menţionat faptul că aceste niveluri de restricţionare a accesului pot fi combinate, ele privind întreaga comunitate a organizaţiei respective (de exemplu, studenţi, cadre didactice, personal auxiliar etc.).

• **WebSearch**

Fiecare sit Web trebuie să ofere utilizatorilor posibilităţi de căutare internă, conform unor criterii diverse. Acest modul va fi implementat astfel încât să poată fi folosit pentru orice tip de sit Web, oferind:

- căutări bazate pe termeni (expresii), folosind inclusiv operatorii utilizaţi de motoarele de căutare: AND, OR, NEAR etc.;
- posibilitatea folosirii expresiilor regulate (în stilul Perl) în cadrul interogărilor;
- căutări ierarhice, bazate pe arbori de termeni – *directories* (de exemplu, *Facultate :: Activitate :: Catedre :: Sisteme Software*).

• **WebDoc**

Să se creeze o aplicaţie Web care să ofere accesul la conţinut documentar, structurat pe teme şi subteme de interes (de exemplu, *Programare :: Limbaje :: Scripting :: Perl* ori *IA :: Recunoaştere forme :: Reţele neurale :: Auto-învăţare*), conform mecanismelor de autentificare şi acordare a accesului oferite de *WebIdent* şi *WebRights*. Se va avea în vedere faptul că documentaţiile:

- pot fi stocate în diverse formate (proprietare sau nu), în manieră distribuită (pe mai multe maşini);
- vor putea fi accesibile în limbi multiple (de exemplu engleză şi/sau română);
- vor putea fi de un anumit tip (carte, manual de referinţă, specificaţie, tutorial etc.), într-un anumit stadiu (ciornă, propus pentru standardizare, final etc.).

Pentru fiecare resursă, se vor reţine cel puţin următoarele date:

- autor(i);
- titlu;
- tip (carte, articol, raport tehnic, referat etc.);
- editor (de exemplu, Springer Verlag, Prentice-Hall,...);
- an al apariţiei;
- loc al apariţiei;
- rezumat;
- cuvinte-cheie (*i.e.* Web, XML, căutare structurată, DOM);
- legătură spre documentul complet (dacă există) sau spre o oglindire locală.

Situl **WebDoc** va putea oferi statistici privind cele mai accesate documente, localizarea clienţilor şi va oferi posibilitatea de a fi administrat via Web, de la distanţă. De asemenea, se va folosi un motor de căutare internă, în cadrul sitului – a se vedea *WebSearch*.

• **WebLex**

Activitatea unei organizaţii academice, în general, şi a unei facultăţi, în special, este guvernată de diverse acte legislative (*i.e.* legi şi prevederi). Acestea pot avea caracter:

- general (de exemplu, legi, hotărâri guvernamentale etc.);
- local (*i.e.* pe teritoriul unei localităţi, hotărâri ale Consiliului Municipal),
- universitar (de exemplu, hotărâri ale Senatului);
- specifice facultăţii (de exemplu, prevederi ale Consiliului facultăţii referitoare la acordarea burselor).

De remarcat faptul că unele acte legislative nu vor putea fi accesate decât de anumite categorii de utilizatori (de pildă, unele reglementări interne nu privesc decât cadrele didactice şi nu studenţii).

Situl **WebLex** va trebui să ofere informaţii despre toate aceste legi şi prevederi, cu posibilitatea căutării – prin intermediul modulului *WebSearch* – şi a restricţionării accesului (de exemplu, persoanele din afara organizaţiei nu au dreptul de a parcurge anumite regulamente interne) – vezi *WebRights*.

• **WebPages**

Pentru a asigura uniformitatea afişării conţinutului paginilor personalului organizaţiei, se va implementa un modul care va realiza o serie de şabloane (*template*-uri) pentru paginile personale ale membrilor. Fiecare şablon va oferi posibilităţi de structurare, vizualizare şi actualizare a informaţiilor cel puţin în limbile română şi engleză. De exemplu, în cazul unei facultăţi, informaţiile importante vor putea fi cele legate de:

- posibilităţi de contact;
- studii efectuate (licenţă, masterat, doctorat etc.);
- activitatea ştiinţifică (listă de publicaţii, conferinţe, proiecte de cercetare etc.) – vezi aplicaţia *WebPub*;
- activitatea didactică (discipline predate, manuale, conducere cercuri studenţeşti şi lucrări de diplomă etc.) – cu legături către paginile disciplinelor (vezi *WebCourses*);
- activitatea managerială (membru în consiliu, funcţii de conducere, membru în comisii de licenţă etc.);
- pasiuni;
- altele.

Se vor oferi posibilităţi de căutare şi de restricţionare a accesului la diferite informaţii.

• **WebPub**

Apare necesitatea de a organiza şi administra lista publicaţiilor ştiinţifice ale organizaţiei (de exemplu, rapoartele tehnice) sau ale unui cadru didactic (de exemplu, lista cărţilor publicate). Acest modul va da posibilitatea de a stoca, întreţine, interoga şi vizualiza informaţiile privitoare la publicaţii printr-o interfaţă Web:

- autor(i);
- titlu;

- tip (carte, articol, raport tehnic etc.);
- editor (de exemplu, O'Reilly & Associates);
- an al apariţiei;
- loc al apariţiei;
- rezumat;
- cuvinte-cheie (*i.e.* servicii Web, XML, XML-RPC, SOAP);
- legătură spre documentul complet (dacă există).

Acest sit se poate folosi de funcţionalităţile oferite de *WebSearch*, *WebRights*, *WebIdent* şi *WebDoc*.

• **WebCourses**

Asemănătoare cu *WebPages*, acestă aplicaţie îşi propune să realizeze o serie de şabloane (*template*-uri) pentru paginile dedicate unei anumite discipline. Fiecare şablon va oferi posibilităţi de structurare, vizualizare şi actualizare a informaţiilor cel puţin în limbile română şi engleză. Informaţiile importante vor putea fi cele legate de:

- scurta prezentare a disciplinei, relaţiile cu alte discipline (graful de dependenţă; de exemplu „Sisteme de Operare II" depinde de „Sisteme de Operare I" şi de „Programare I");
- programa disciplinei (*curriculum*);
- anul şi semestrul de studii (de exemplu, anul III, semestrul 2), numărul de ore alocate pentru curs/laborator/seminar;
- tipul disciplinei (obligatorie, opţională, facultativă);
- persoanele implicate: titularul/titularii de curs şi alte cadre didactice susţinând seminarii/laboratoare;
- cerinţele specifice fiecărei discipline (modul de promovare, proiecte, lucrări pe parcurs, teme pentru acasă etc.);
- orarul/locul de desfăşurare a cursului, laboratorului sau seminarului;
- modalitatea de înscriere la curs;
- organizarea resurselor oferite studenţilor: prezentări, documentaţii (a se vedea *WebDoc*), cărţi şi articole (vezi şi aplicaţia *WebPub*);
- relaţia profesor-student: listă de discuţii, forum, *chat*, alte posibilităţi de contact etc.;
- modalitatea de testare a cunoştinţelor via aplicaţia *WebTest*;
- modalitatea de înscriere la examen – vezi *WebExam*;
- orarul/locul de desfăşurare a examenului sau examenelor la acea disciplină.

• **WebExam**

Oferă posibilitatea studenţilor să se înscrie la un examen la o dată diferită de cea programată la grupa din care fac parte. Un anumit student, pe baza identificării (conform funcţionalităţilor implementate de *WebIdent*), va putea alege data (datele) la care doreşte să susţină examenele programate în perioada de sesiune sau în cea de restanţe. Această înscriere poate fi dictată de anumiţi termeni impuşi de titularul disciplinei ori de secretariat (de exemplu, un număr maxim de studenţi, înscrierea să se efectueze cu *N* zile înainte de data programată pentru examen sau alte condiţii de blocare a înscrierii).

Se va da posibilitatea administrării informaţiilor despre un anumit examen (nume disciplină, grupe, data susţinerii, număr maxim de studenţi admişi la proba de examen etc.) şi generării de rapoarte – eventual grafice – privitoare la studenţii înscrişi (pe ani şi grupe de studii, pentru fiecare disciplină în parte).

• **WebTest**

Să se implementeze o aplicaţie pentru stocarea în manieră distribuită a diferitelor tipuri de teste de cunoştinţe care pot fi folosite în regim de testare sau auto-testare la diverse discipline (în conjuncţie cu *WebCourses*).

Pe baza autentificării (folosind *WebIdent*), titularii de curs vor putea gestiona colecţia de teste de la discipline diferite (accesare întrebări şi răspunsuri, realizare de teste prin alegerea întrebărilor după diverse criterii sau aleatoriu etc.).

Studenţii vor putea avea acces la module de auto-testare pe diverse domenii, contra-cronometru sau nu (*demo*-uri ale unor teste reale în regim de examen).

Testele vor putea fi de mai multe tipuri (*single/multiple choice*, cu diferite modalităţi de acordare a punctajelor, cu întrebări dependente una de cealaltă etc.), bazate sau nu pe cronometrare. Aplicaţia va putea oferi rapoarte privitoare la prestaţia unui anumit student la o disciplină, eventual diferite statistici – numerice şi grafice – referitoare la gradul de cunoştinţe ale studenţilor (de exemplu, numărul răspunsurilor eronate, timpul realizat, întrebările la care n-a răspuns nici unul dintre studenţii testaţi etc.).

Se vor oferi posibilităţi de căutare de teste pe diverse criterii – domeniu de interes (*i.e.* limbajul Java), cuvinte-cheie (de exemplu, moştenire multiplă), grad de dificultate etc. (vezi şi *WebSearch*) – şi de restricţionare a accesului prin *WebRights* (de exemplu, doar studenţii înscrişi la cursul de programare algebrică vor beneficia de [auto-]testare la acest obiect).

• **WebDesignChecker**

Să se realizeze o aplicaţie care să verifice dacă un anumit sit respectă principiile de design şi de ergonomie Web. Aplicaţia va putea oferi şi sugestii privind corectarea diverselor erori de design întâlnite în cadrul documentelor (de exemplu, validitatea documentelor, legături inexistente, documente de dimensiuni prea mari, utilizarea prea multor fonturi/culori într-o pagină, lipsa de mijloace de navigare prin paginile sitului etc.).

ANEXA A

Detalii de instalare

Bogdan Manolache, toni@MVDmedia.ro
Sabin Buraga, busaco@infoiasi.ro

În această anexă vom prezenta paşii care trebuie urmaţi pentru a instala pe propriul calculator aplicaţiile necesare experimentării programării Web în PHP.

Pentru început ne vom concentra asupra instalării, pe o platformă Windows XP, a serverului Web *IIS* (*Internet Information Services*), a serverului de baze de date MySQL şi a serverului de aplicaţii PHP. Nu este obligatoriu să folosiţi IIS drept server Web, alternativa fiind desigur Apache, dar IIS este inclus în *kit*-ul sistemului de operare Windows XP Professional, trebuind doar să fie pornit ca serviciu de sistem.

Pentru instalare vor trebui urmaţi paşii de mai jos:

1. Se instalează serverului IIS şi extensiile FrontPage:
- deschideţi **Control Panel** (*Start* → *Control Panel*);
- deschideţi **Add or Remove Programs**;
- selectaţi **Add/Remove Windows Components** din partea stângă a ferestrei;
- selectaţi apoi **Internet Information Services (IIS)**;
- apăsaţi pe **Details** şi asiguraţi-vă că opţiunea **FrontPage < version > Server Extension** este selectată;
- având *kit*-ul sistemului de operare Windows XP la îndemână apăsaţi **OK**, apoi **Next**;
- va apărea o fereastră în care trebuie să indicaţi locaţia exactă a *kit*-ului de Windows XP – de cele mai multe ori unitatea de CD;
- apăsaţi butonul **Finish**.

2. Se configurează extensiile FrontPage (*FrontPage Server Extensions*):
- deschideţi **Control Panel** (*Start* → *Control Panel*);
- deschideţi **Administrative Tools**;
- deschideţi **Computer Management**;
- în fereastra care se va deschide extindeţi ramura **Services and Applications**;
- extindeţi ramura **Internet Information Services**;
- extindeţi ramura **Web Sites**;
- apăsaţi *click* dreapta pe **Default Web Site**, iar din meniul care se deschide selectaţi apoi **All Tasks** → **Configure Server Extensions**;
- în fereastra care se deschide apăsaţi butonul **Next**, apoi **YES**, **Next** şi **Finish**.

Am terminat instalarea serverului Web. Pentru a testa dacă tot ce am făcut până acum funcţionează, vom deschide navigatorul Web preferat şi vom introduce ca adresă `localhost`. Dacă totul funcţionează, se vor deschide două ferestre una dintre ele având ca titlu **Welcome to Windows XP Server Internet Services**, iar cealaltă conţinând un manual pentru IIS. Dacă nu se deschide nici o fereastră şi apare mesajul **The page cannot be displayed** atunci repetaţi paşii descrişi mai sus, începând de la punctul 1.

3. Se instalează serverul de baze de date MySQL:

Dacă până acum nu am avut nevoie de nici o resursă suplimentară (exceptând *kit*-ul de Windows XP), pentru instalarea MySQL trebuie procurat *kit*-ul de instalare. Fiind disponibil gratuit, îl putem găsi la adresa **www.mysql.com**, secţiunea *Download*. Varianta curentă are aproximativ 14M. După ce aţi dezarhivat programul de instalare de la MySQL rulaţi **Setup.exe** şi instalaţi serverul. În mod uzual, MySQL se va instala în directorul `c:\mysql`.

4. Se instalează serverul de aplicaţii PHP:

Aceasta este, de fapt, etapa cea mai importantă, fără de care nu veţi putea testa nici o bucată de cod PHP din carte. Aveţi nevoie de *kit*-ul de instalare pe care puteţi să-l luaţi de la adresa **www.php.net**, secţiunea *Downloads*. După ce programul de instalare a fost transferat pe calculatorul dumneavoastră, rulaţi-l. Veţi avea de ales între instalarea **Standard** şi instalarea **Advanced**. Selectaţi a doua opţiune, **Advanced**. Câţiva paşi mai încolo alegeţi tipul serverului Web, în cazul nostru **Microsoft IIS 4 or higher**. Va apărea o fereastră **IIS Scriptmap Node Selection** în care bifaţi toate căsuţele apăsând pe **Select All**. Am terminat astfel şi instalarea serverului PHP.

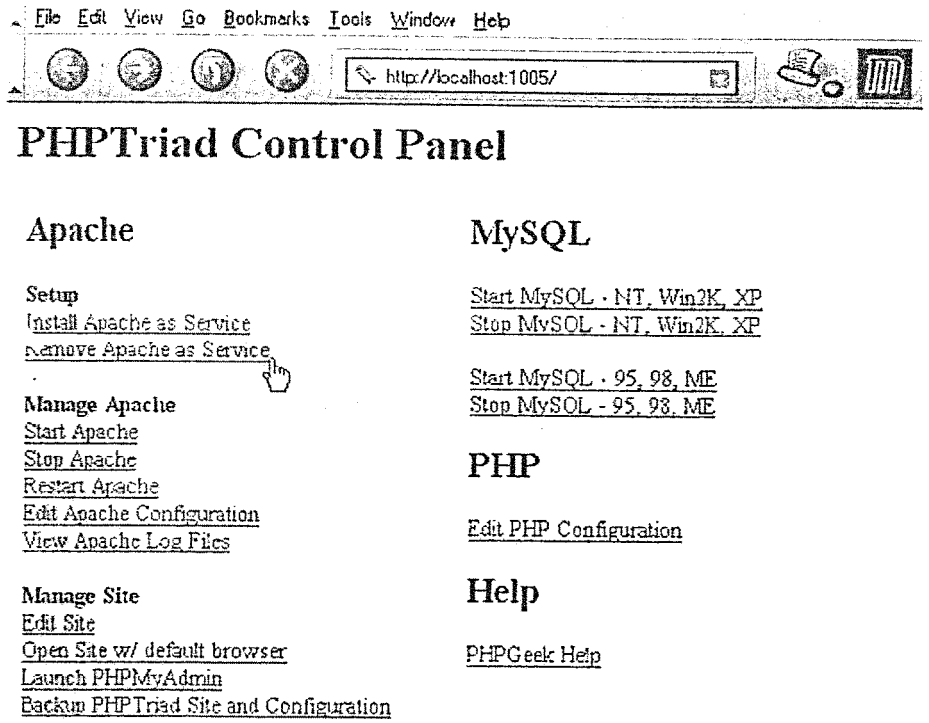
Pentru a verifica dacă instalarea s-a efectuat corect, editaţi un fişier text cu ajutorul oricărui editor de texte (*Notepad*, de exemplu). În acest fişier scrieţi următoarele rânduri:

```
<?php
   phpinfo(); // afiseaza informatii despre PHP
?>
```

Salvaţi fişierul cu numele `test.php` şi copiaţi-l în directorul `c:\Inetpub\wwwroot\`. Deschideţi un navigator şi introduceţi adresa `http://localhost/test.php`. Dacă totul a fost instalat în bune condiţii, atunci ar trebui să apară o pagină Web cu informaţii referitoare la PHP, generată de funcţia predefinită `phpinfo()`. Printre informaţiile utile afişate se numără versiunea curentă a serverului PHP, extensiile disponibile, suportul pentru procesarea diverselor tipuri de documente etc.

O alternativă la cele de mai sus este aceea a utilizării aplicaţiei *PHPTriad* care va instala – pe un sistem Windows 98, Me, 2000 sau XP – dintr-un singur *click* cele trei componente majore de care avem nevoie: serverul Web Apache, serverul de baze de date MySQL şi serverul de aplicaţii PHP. Programul include şi aplicaţia de administrare comodă a bazelor de date *phpMyAdmin*. Tot ceea ce trebuie făcut este să vă procuraţi cea mai recentă versiune *PHPTriad* de la adresa următoare: **http://sourceforge.net/projects/phptriad** (versiunea 2.2.1 este disponibilă pe CD-ul cărţii *Proiectarea siturilor Web* de Sabin Buraga). Avizăm cititorul că *PHPTriad* nu include cele mai recente şi sigure versiuni ale serverelor mai sus amintite şi, aşadar, poate fi utilizată doar cu scopul de învăţare a programării Web sau de testare a unor programe PHP şi nu pentru a menţine un sit Web public. Varianta comercială a aplicaţiei *PHPTriad* se numeşte *Sokkit* şi poate fi încercată transferând-o de pe situl **http://www.sokkit.net**.

După instalare, directorul `c:\apache` va găzdui toate „ingredientele" de care avem nevoie. Pentru a activa serverul Apache, vom folosi *Start* → *Programs* → *PHPTriad* → *Apache Console* → *Start Apache*. Va fi deschisă o fereastră MS-DOS în care vor apărea unele mesaje ale serverului Apache şi pe care o putem minimiza. Pentru a închide sau restarta Apache, în loc de *Start Apache* vom utiliza *Stop Apache* şi, respectiv, *Restart Apache*. După ce am făcut operaţional serverul Apache, vom rula şi serverul MySQL prin *Start* → *Programs* → *PHPTriad* → *MySQL* → *MySQL-D* (sau *MySQL-D-NT*). Pentru Windows XP avem posibilitatea ca MySQL să ruleze ca serviciu (a se vedea *Administrative Tools* → *Services*). Vom testa dacă totul a fost instalat în bune condiţii procedând ca mai sus. Fişierul `test.php` va fi plasat de această dată în directorul `c:\apache\htdocs`, apoi vom lansa un navigator Web şi vom tasta adresa `http://localhost/test.php`. Pentru a utiliza aplicaţia *phpMyAdmin* (amintită şi în capitolul 1) vom folosi adresa `http://localhost/phpmyadmin`.



Administrarea serverelor cu *PHPTriad Control Panel*

În ceea ce priveşte un sistem Linux, orice distribuţie actuală include Apache, MySQL şi PHP care pot fi instalate chiar la momentul instalării sistemului de operare. În cazul popularelor distribuţii RedHat şi Mandrake sunt puse la dispoziţie pachetele `.rpm` necesare instalării serverelor amintite pe CD-urile de instalare (puteţi folosi şi un instrument grafic de instalare precum *GnoRPM* din managerul de ferestre GNOME).

ANEXA B

Limbajul PHP pe scurt

Sabin Buraga, busaco@infoiasi.ro

Limbajul PHP este un limbaj de tip *script*, procedural, oferind şi suport pentru programarea orientată-obiect. Programele PHP (stocate în mod uzual de fişiere cu extensia .php) vor fi accesibile direct din cadrul paginilor Web. Sintaxa limbajului este inspirată din cea a limbajelor C/C++ (a se consulta, de exemplu, *Programare C şi C++ pentru Linux* scrisă de D. Acostăchioaie) şi Perl (vezi *Programare Web în bash şi Perl* de S. Buraga *et al.*).

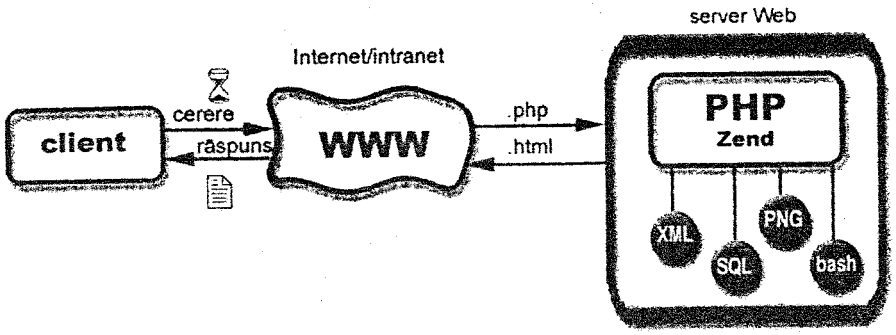
Puterea limbajului constă în setul cuprinzător de funcţii puse la dispoziţie în special pentru manipularea bazelor de date într-un dialect SQL (sunt suportate diverse servere de baze de date precum MySQL, Oracle, PostgreSQL, Sybase etc.) sau pentru procesarea conţinutului unei largi palete de tipuri de fişiere (ca de exemplu, diferite tipuri de arhive, documente PDF, fişiere grafice, documente XML, certificate digitale etc.). Aceste funcţii pot fi grupate pe mai multe categorii (o serie fiind similară funcţiilor din bibliotecile standard ale limbajului C), dintre care se pot menţiona: matematice, de conversie de tipuri, de manipulare a şirurilor de caractere, de prelucrare a tablourilor, de acces la resursele sistemului de operare şi de lucru cu fişiere, de manipulare a bazelor de date, privitoare la conexiunile Internet (de exemplu, HTTP, FTP, IMAP etc.), pentru accesarea fişierelor având tipuri speciale: PDF, XML, gzip etc., specifice sistemului de operare, generale.

Scripturile PHP pot fi incluse în paginile Web în diferite moduri, uzual folosindu-se construcţia <?php ... ?>. Comentariile pot fi scrise în stilul comentariilor din limbajele C/C++ (/* ... */ sau //...) ori Perl (#...). Declaraţiile şi instrucţiunile se separă prin „;" ca şi în C, Perl sau Java.

Funcţionarea analizorului (motorului) PHP este următoarea: preprocesorul PHP va determina construcţiile PHP delimitate între *tag*-urile <?php şi ?> şi le va interpreta, ignorând restul conţinutului documentului pe care îl va trimite, la ieşirea standard, nealterat. Astfel, putem privi analizorul PHP ca un script CGI care efectuează o preprocesare specială înainte ca datele să fie expediate clientului Web. PHP se bazează pe motorul de *scripting* Zend.

Tipurile de date suportate sunt cele simple (scalare): logic (*boolean*), întreg (*integer*), flotant (*float*, *double*, *real*) şi şir de caractere (*string*). Tipurile compuse sunt reprezentate din tipul tablou (*array*) – indexat şi asociativ – şi din tipul obiect (*object*). Mai sunt disponibile şi două tipuri speciale: tipul resursă (*resource*) şi nul (*NULL*).

Menţionăm că şirurile de caractere se pot delimita între ghilimele sau apostrofuri, în acest ultim caz inhibându-se evaluarea variabilelor din interiorul şirului respectiv (exact ca la *script*-urile bash ori Perl).



Funcţionarea motorului PHP

Variabilele în PHP se reprezintă prin intermediul unui identificator prefixat de caracterul „$". Numele variabilei este *case-sensitive* (*i.e.* variabila `$PHP` este diferită de `$php`) şi e compus din caractere alfa-numerice şi simbolul „_". Tipul unei variabile se determină în funcţie de valoarea pe care o stochează. Variabilele se pot folosi fără a trebui să fie declarate în prealabil. Există, de asemenea, şi diverse *variabile predefinite*, mai ales cele specifice serverului Web. Tot ca variabile sunt transmise unui script PHP câmpurile formularelor HTML dintr-o pagină sau *cookie*-urile. Un tip special sunt variabilele globale (*globals*) care stochează diverse categorii de variabile externe (de exemplu, `$_GET`, `$_POST`, `$_COOKIES`, `$_ENV` etc.).

Constantele pot fi considerate variabile ale căror valori sunt fixate şi nu mai pot fi alterate pe parcursul programului. Pentru a defini o constantă se va folosi construcţia `define()`. Ca exemple de *constante predefinite* se pot da `PHP_VERSION`, `PHP_OS` sau `CHAR_MAX`.

PHP pune la dispoziţie un set impresionant de *operatori*. Aceşti operatori sunt cei disponibili în limbaje precum C sau Perl. Categoriile importante de operatori sunt: aritmetici, de asignare (ca de exemplu „+="), pe biţi, relaţionali (ca noutate se poate menţiona aici „===" care are ca rezultat constanta `TRUE` dacă operanzii au valori identice şi în plus sunt de acelaşi tip), de control al erorilor (există un singur operator „@" care inhibă erorile ce pot apărea în cadrul expresiei pe care o prefixează), de incrementare/decrementare („++" şi „--"), logici (`and` `or` sau „!", „&&" şi „||"), pentru şiruri (se defineşte operatorul de concatenare „." similar celui din Perl). Poate fi folosit, de asemenea, şi operatorul de test „?:".

Instrucţiunile sunt cele uzuale din alte limbaje de programare. Sunt disponibile instrucţiuni de asignare (de forma `$variabilă = expresie`), instrucţiuni de test (`if` şi `switch`), instrucţiuni repetitive (`while`, `do`, `for`, `foreach`) şi instrucţiuni de control (`break`, `continue`, `return`).

Un exemplu de afişare a sumei valorilor unui tablou unidimensional (vector), prin utilizarea variabilelor de tip întreg şi tablou indexat, a instrucţiunilor `for` şi `foreach`, a operatorilor relaţionali, de asignare şi de incrementare (funcţia predefinită `echo()` afişează la intrarea standard – aici fereastra navigatorului – codul HTML generat):

```
<?php
   // umplem un tablou cu valori de la 1 la 10
   for ($contor = 1; $contor <= 10; $contor++) {
     $valori[$contor] = $contor;
   }

   // realizam suma valorilor, utilizand foreach
   $suma = 0;
   foreach ($valori as $element)
     $suma += $element;

   // afisam suma obtinute la iesirea standard
   // pentru a fi trimisa browserului - folosim echo()
   echo ("<p>Suma de la 1 la 10 este: <b>"
         . $suma . "</b></p>");
?>
```

Funcţiile pot fi predefinite sau definite de utilizator. Cele definite de utilizator se specifică printr-o construcţie de genul:

```
function nume ($arg_1, $arg_2, ..., $arg_n)
{
    // instructiuni
    return $valoare; /* optional se returneaza o valoare */
}
```

Apelul unei funcţii se realizează prin numele său, furnizând eventual şi valorile pentru argumentele sale: `nume (expr_1, expr_2, ..., expr_n)`. Pentru un argument se poate furniza o valoare implicită, ca în C++.

În următorul exemplu definim funcţia *patrat()* care calculează pătratul unui număr întreg. Vom folosi această funcţie pentru afişarea în fereastra navigatorului a pătratelor primilor 5 întregi naturali pari:

```
<body text="white" bgcolor="black">
<?php
// functia de ridicare la patrat
function patrat($numar) {
  return $numar * $numar;
}

$numar = 0;

while ($numar < 10) {
  $numar++;          // incrementam numarul

  if ($numar % 2) // e numar impar...
    continue;        // continuam cu urmatoarea iteratie

  // e numar par, deci afisam patratul
```

```
  echo "<p>$numar la patrat este " . patrat($numar) . "</p>";
}
?>
</body>
```

Limbajul PHP, mai ales de la versiunea PHP 4, oferă implementarea unora dintre cele mai importante aspecte ale programării obiectuale: încapsularea datelor, moştenirea şi polimorfismul. Suportul pentru programarea orientată-obiect se realizează prin intermediul *claselor*.

Datele membre se specifică utilizând construcţia `var`, putând avea orice tip (de exemplu, întreg, tablou sau chiar obiect). Metodele se definesc prin declaraţii de funcţii care prelucrează membrii clasei. Pentru a accesa datele clasei, metodele vor trebui să se folosească de construcţia sintactică – inspirată de C++ ori Java – `$this->variabila`, altfel `variabila` se consideră a fi locală în cadrul acelei metode. Se pot specifica, de asemenea, constructori ai unei clase.

Instanţierea unui obiect al clasei se realizează prin operatorul `new`, exact ca în limbajul C++. PHP pune la dispoziţie suport pentru moştenire, o clasă putând fi derivată dintr-alta prin intermediul construcţiei sintactice `extends`.

Un exemplu de definire a unei clase şi de instanţiere a unui obiect al acesteia:

```
<?php

// definim o constanta
define("AN_IMPLICIT", 1);

class Student {

  // date-membru
  var $an;          // anul de studii
  var $nume;        // numele
  var $email;       // adresa de e-mail

  // constructor
  function Student($a = AN_IMPLICIT, $n = '', $e = '') {
    $this->an = $a;
    $this->nume = $n;
    $this->email = $e;
  }

  // metode
  function seteazaAn($un_an) {
    $this->an = $un_an;
  }
  function furnizeazaAn() {
    return $this->an;
  }
```

```
  // alte metode...
}

// instantierea unui obiect al clasei Student
// ultimul argument va avea valoarea implicita
$stud = new Student (4, "Bogdan Manolache");
// utilizarea datelor membre si a metodelor clasei
$nume = $stud->nume;
echo ("<p>Anul de studii al lui <b>$nume</b> este " .
      $stud->furnizeazaAn() . "</p>");
// verificam daca a fost furnizata adresa de e-mail
if (!$stud->email) {
  echo ("<p>Adresa de e-mail nu a fost setata.</p>");
}
?>
```

O clasă derivată a clasei de mai sus se poate defini astfel:

```
class StudentDestept extends Student {
  // date-membru suplimentare
  var $note;    // un tablou cu notele obtinute

  // metode suplimentare
  function seteazaNote($niste_note) {
    $this->note = (array) $niste_note;
  }
  function furnizeazaNote() {
    return (array) $this->note;
  }
}
```

Vom instanţia un obiect din clasa derivată (se va apela constructorul clasei de bază) şi vom utiliza metodele noi definite:

```
// un tablou asociativ continand notele
$note2003 = array ("Web" => 10, "RC" => 10, "LFA" => 9);

// instantierea unui obiect din clasa StudentDestept
$alt_student = new StudentDestept (3,
                        "Iavi Rotberg", "pit@infoiasi.ro");
$alt_student->seteazaNote($note2003);
// preluam notele din obiect...
$alte_note = $alt_student->furnizeazaNote();
// ...si le comparam cu cele din tabloul note2003
if ($note2003 === $alte_note) {
  echo ("<p>Notele au ramas aceleasi...<br />");
```

```
  echo ("La \"Tehnologii Web\", " .
        "$alt_student->nume a luat <b>" .
        $alte_note["Web"] . "</b>.</p>");
}
else
  echo ("<p>Notele s-au schimbat!?</p>");
```

Pentru mai multe amănunte privind limbajul, recomandăm consultarea referinţelor bibliografice şi a manualului PHP.

ANEXA C

Procesarea documentelor XML în PHP

Sabin Buraga, busaco@infoiasi.ro

Această anexă pune la dispoziţie codul complet al unei clase PHP care încapsulează activitatea de procesare a documentelor XML, folosind modelul *SAX* (*Simple API for XML*) via procesorul *Expat*.

Clasa este implementată de următorul fişier (denumit parseXML.php):

```
<?php

// o clasa pentru prelucrarea documentelor XML

class parseXML {
  var $xml_parser; /* instanta analizorului XML */
  var $xml_file;   /* numele fisierului XML */
  var $html_code;  /* codul (X)HTML generat */
  var $open_tags;  /* multimea tag-urilor de inceput */
  var $close_tags; /* multimea tag-urilor de sfarsit */

  // constructor
  function parseXML() {
    $this->xml_parser = "";
    $this->xml_file = "";
    $this->html_code = "";
    $this->open_tags = array();
    $this->close_tags = array();
  }
  // destructor
  function destroy() {
    if ($this->xml_parser)
      xml_parser_free($this->xml_parser);
  }

  // metode
  // seteaza tag-urile de inceput
```

```
function set_open_tags($tags) {
  $this->open_tags = $tags;
}
// seteaza tag-urile de sfarsit
function set_close_tags($tags) {
  $this->close_tags = $tags;
}
// seteaza numele fisierului XML
function set_xml_file($file) {
  $this->xml_file = $file;
}
// furnizeaza codul (X)HTML generat
function get_html_code() {
  return $this->html_code;
}
// tratarea evenimentului de
// aparitie a unui tag de inceput
function start_element($parser, $name, $attrs) {
  if ($format = $this->open_tags[$name])
    $this->html_code .= $format;
}
// tratarea evenimentului de
// aparitie a unui tag de sfarsit
function end_element($parser, $name) {
  if ($format = $this->close_tags[$name])
    $this->html_code .= $format;
}
// tratarea evenimentului de
// aparitie a unui CDATA
function character_data($parser, $data) {
  $this->html_code .= $data;
}
// tratarea evenimentului de
// aparitie a unei instructiuni de procesare
function processing_instruction($parser, $target, $data) {
  switch (strtolower($target)) {
    case "php": eval($data);
                break;
  }
}
// functia de analiza propriu-zisa
function parse() {
  // instantiaza procesorul XML
  $this->xml_parser = xml_parser_create();
  // inregistreaza functiile de analiza
```



```
    xml_set_object($this->xml_parser, &$this);
    // seteaza optiunile de analiza
    // (tag-urile nu sunt rescrise cu caractere mari)
    xml_parser_set_option($this->xml_parser,
                          XML_OPTION_CASE_FOLDING, false);
    // seteaza functiile de procesare a elementelor XML
    xml_set_element_handler($this->xml_parser,
                            "start_element", "end_element");
    xml_set_character_data_handler($this->xml_parser,
                            "character_data");
    xml_set_processing_instruction_handler($this->xml_parser,
                            "processing_instruction");

    // deschide fisierul XML
    if (!($fp = fopen($this->xml_file, "r")))
      die("could not open XML source");
    // proceseaza fisierul
    while ($data = fread($fp, 4096)) {
      if (!xml_parse($this->xml_parser, $data, feof($fp))) {
        // eroare de procesare
        die(sprintf("XML error: %s at line %d",
          xml_error_string(xml_get_error_code
($this->xml_parser)),
          xml_get_current_line_number($this->xml_parser)));
      }
    } /* while */
  } /* parse() */
} /* class */
?>
```

Sunt descrise în continuare două exemple de utilizare. Intenţionăm să procesăm următorul document XML – stocat în fişierul `impr.xml` – adnotând impresii lăsate de vizitatorii unui sit Web:

```
<impresii>
  <impresie data="04 Sep 2003, 18:42"
            client="localhost (127.0.0.1)">
  <nume email="alexo@infoiasi.ro">Alexandrina Orzan</nume>
  <ocupatia>Absolvent</ocupatia>
  <virsta>25-35</virsta>
  <text>
Un prim mesaj...
  </text>
  </impresie>
  <impresie data="03 Sep 2003, 10:10"
            client="fenrir.infoiasi.ro (193.231.30.197)">
```

```
  <nume email="toni@infoiasi.ro">Bogdan Manolache</nume>
  <ocupatia>Student</ocupatia>
  <virsta>20-25</virsta>
  <text>
Am citit, acum implementez!
  </text>
  </impresie>
</impresii>
```

Primul script procesează elementele XML ale documentului, instanţiind un obiect al clasei mai sus definite:

```
<?php
  // exemplu de procesare XML folosind PHP si Expat

  require("parseXML.php");

  // substitutia elementelor XML cu cod XHTML
  // se folosesc doua tablouri asociative
  $open_tags = array(
    "impresii" => "\n<!-- generat de parseXML -->\n" .
       "<table cellpadding=\"5\" align=\"center\"
border=\"1\" summary=\"continut\">",
    "impresie" => "<tr align=\"center\">",
    "nume" => "<td><h4>",
    "ocupatia" => "<td><p style=\"color: blue\">",
    "virsta" => "<td><p><i>",
    "text" => "<td bgcolor=\"#EEEEEE\"><p align='justify'>");
  $close_tags = array(
    "impresii" => "</table>\n" .
       "<!-- sfirsitul generarii parseXML -->\n",
    "impresie" => "</tr>",
    "nume" => "</h4></td>",
    "ocupatia" => "</p></td>",
    "virsta" => "</i></p></td>",
    "text" => "</p></td>");

  // instantiaza si initializeaza analizorul
  $parser = new parseXML();
  $parser->set_xml_file("impr.xml");
  $parser->set_open_tags($open_tags);
  $parser->set_close_tags($close_tags);
  // ruleaza analizorul
  $parser->parse();
  // afiseaza rezultatul
  echo $parser->get_html_code();
```



```
  // distruge obiectul
  $parser->destroy();
?>
```

Al doilea exemplu oferă un mod de procesare al atributelor documentului XML (în cazul de faţă atributul `email`):

```
<html>
<head>
  <title>Impresii</title>
</head>
<body bgcolor="white" text="black">
<?php

  // exemplu de procesare XML folosind PHP si Expat
  require("parseXML.php");

  // folosirea mostenirii pentru a defini
  // un alt comportament
  class parseXML2 extends parseXML {

    // indica daca exista atributul "email"
    var $is_email = 0;

    // redefinirea metodelor
    function start_element($parser, $name, $attrs) {
      // apeleaza metoda din clasa de baza
      parseXML::start_element($parser, $name, $attrs);
      // pune si link spre adresa e-mail
      if (!strcmp($name, "nume")) {
        if ($attrs["email"]) {
          $format = "<a title=\"Trimite mesaj la " .
                        $attrs["email"] .
                        "\" href=\"mailto:" .
                        $attrs["email"] . "\">";
          $this->html_code .= $format;
          $this->is_email = 1;
        }
        else
          $this->is_email = 0;
      }
    }

    function end_element($parser, $name) {
      // inchide </a>
      if (!strcmp($name, "nume")) {
```

```
        if ($this->is_email) {
          $format = "</a>";
          $this->html_code .= $format;
        }
      }
      // apeleaza metoda din clasa de baza
      parseXML::end_element($parser, $name);
    }
  }

  $open_tags = array(
    "impresii" => "\n<!-- generat de parseXML -->\n" .
       "<table cellpadding=\"10\" align=\"center\"
border=\"1\" summary=\"continut\">",
    "impresie" => "<tr align=\"center\">",
    "nume" => "<td><h4>",
    "ocupatia" => "<td><p style=\"color: blue\">",
    "virsta" => "<td><p><i>",
    "text" => "<td bgcolor=\"#EEEEEE\"><p align='justify'>");
  $close_tags = array(
    "impresii" => "</table>\n" .
       "<!-- sfarsitul generarii parseXML -->\n",
    "impresie" => "</tr>",
    "nume" => "</h4></td>",
    "ocupatia" => "</p></td>",
    "virsta" => "</i></p></td>",
    "text" => "</p></td>");

  // instantiaza si initializeaza analizorul
  $parser = new parseXML2();
  $parser->set_xml_file("impr.xml");
  $parser->set_open_tags($open_tags);
  $parser->set_close_tags($close_tags);
  // ruleaza analizorul
  $parser->parse();
  echo $parser->get_html_code();
  // distruge obiectul
  $parser->destroy();
?>
</body>
</html>
```

ANEXA D

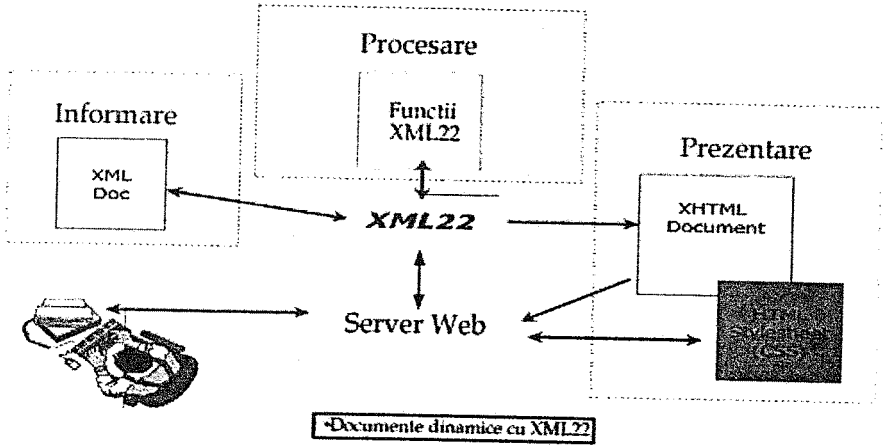
Biblioteca *XML22*

Iavi Rotberg, pit@infoiasi.ro

Pentru a căuta şi edita informaţiile din fişierele XML se va putea folosi biblioteca *XML22* concepută de *Ingo Schramm* – a se vedea adresa **http://www.ister.org/code/xml22/** –, fiind disponibilă în regim *open-source*, sub auspicii GNU General Public License.

Biblioteca, scrisă în PHP, procesează un document XML prin intermediul unui tablou multidimensional. Procesorul XML recunoaşte şi prelucrează orice cod XML, incluzând instrucţiuni de procesare şi entităţi externe. Biblioteca oferă o multitudine de funcţii pentru a procesa, edita şi apoi scrie în memorie sau într-un fişier documentul XML.

Următoarea schemă evidenţiază funcţionarea bibliotecii XML22:



Schema funcţională a bibliotecii XML22

XML22 foloseşte două structuri interne, de tipul tablou asociativ multidimensional, prima pentru a reprezenta orice element (*tag*) din documentul XML, exceptând *tag*-ul „doctype", pentru care se foloseşte a doua structură.

Un tablou de tip „*tag*" are următoarea structură:

```
Array (
    [tag]    =>  string // folosit pentru comentarii
```

```
        [level]   =>  int    // nivelul din arborele XML
        [index] =>  int      // ordinea in cadrul documentului
        [parindex] => int   // referinta spre indexul parintelui
        [children] => Array (
                // referinte spre indicele elementelor-copil
                [int] => int          // document index
                [int] => int          // document index
                )
                // atribute XML
            [attributes] => Array (
                            [name]    => 'value'
                             [name]    => 'value'
                             )
            [content]     =>  string // continut
            ),
    [...]
  )
```

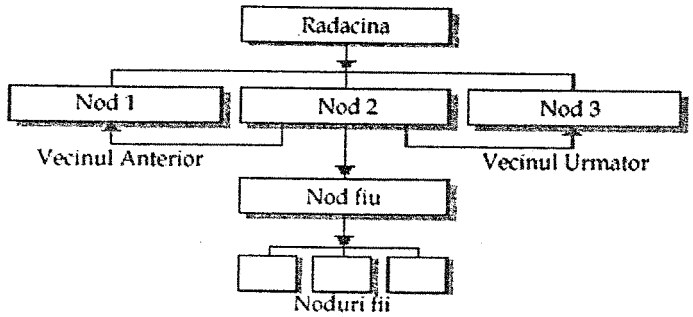
Un tablou de tip *„doctype"* (folosit pentru procesarea definiţiilor de tipuri de documente – DTD) are structura:

```
Array (
[tag] => string  // intotdeauna setat pe "!DOCTYPE"
[level] => int   // nivel, trebuie sa fie 0
[index] => int   // ordinea in cadrul documentului
[type]        =>  Array (
[0] => string    // numele radacinii
[1] => string    // SYSTEM | PUBLIC
[2] => string    // URI daca  SYSTEM, name daca PUBLIC
[3] => string    // URI daca PUBLIC, nu exista daca SYSTEM
)
// entitati XML
[entities]    =>  Array (
// nu exista daca nu exista nici o definitie de tip <!ENTITY
[<string>] => string // nume => definitie
[...]
)
// notatii XML
[notations]   =>  Array (
// nu este setat daca nu exista definitii de tip <!NOTATION
[<string>] => string // nume => definitie
[...]
)
// comentarii XML
// toate comentariile din <!DOCTYPE
[comments]    =>  Array (
```



```
[int] => string
)
// instructiuni de procesare XML
[pi] =>  Array (   // instructiuni de procesare din <!DOCTYPE
[int] => string
)
)
```

Următoarea ierarhie prezintă schematic modul de tratare a unui nod XML de către biblioteca XML22:



Ierarhia nodurilor prin prisma XML22

Ordinea *tag*-urilor din tablou reprezintă ordinea nodurilor din document, dar indicii *tag*-urilor nu sunt echivalenţi cu nivelurile logice din document. Nodul rădăcină al documentului nu are de obicei indexul 0 în tablou. Pentru documente XML bine formatate în elementul cu indicele 0 se găseşte versiunea XML a documentului (*XML version string*). Pentru a obţine nodul rădăcina se foloseşte funcţia: `xml22_get_root()`.

Căutarea şi editarea documentului XML se pot face pe trei niveluri. Astfel documentul poate fi accesat la nivelul abstract XML, la nivelul marcatorilor sau chiar la nivelul reprezentărilor interne.

De exemplu, dacă avem nevoie să extragem dintr-un document XML elementul `<titlu>` al cărui conţinut să fie „28 Ianuarie" îl putem căuta în următoarele moduri:

- la nivel XML:

```
$doc = xml22_parse('noutati.xml');
$firstresult = xml22_get_all_of_name($doc, 'titlu');
$nextresult = xml22_get_all_of_content
                    ($firstresult, '/28 Ianuarie/');
```

- la nivel de *tag*:

```
$doc = xml22_parse('noutati.xml');
$search = array
            ('tag'=>'/titlu/', 'content'=>'/28 Ianuarie/');
$result = xml22_get_by_regex($doc, $search, XML22_GET_FIRST);
```

Un fragment de document (*document fragment*) este un document XML sau o parte a unui document XML fără declaraţia de început `<?xml` şi fără a avea asociat nici un DTD. Lucrând cu fragmente, viteza scripturilor va creşte. De exemplu, codul de mai jos afişează fragmentul cu toate *tag*-urile care aparţin primului copil al rădăcinii din arborele asociat documentului XML:

```
$doc = xml22_parse('file.xml');
$parent = xml22_get_first_child($doc,xml22_get_root($doc));
$offset = count(xml22_get_all_descendants($doc, $parent));

$frament = xml22_copy_fragment($doc, $parent, $offset);
print xml22_write_document_str($fragment, XML22_BEAUTIFY);
```

Uneori este posbil ca scripturile să încerce să citească acelaşi document XML de mai multe ori, de exemplu în cazul unor documente concatenate care pointează unele spre celelalte. Pentru a preveni acest lucru, funcţia care procesează documentul încearcă să introducă datele deja analizate într-o memorie *cache*. Acest lucru conduce la creşterea performanţei. De asemenea, mecanismul de *cache* verifică dacă documentul a fost modificat de la ultima procesare.

Un exemplu:

```
$doc1 = xml22_parse("file1.xml");
// file1 este procesat si introdus in memoria cache
unset($doc1);
$doc2 = xml22_parse("file2.xml");
// file2 este procesat si introdus in memoria cache
unset($doc2);
$doc1 = xml22_parse("file1.xml");
// file1 este regasit in cache si nu va mai fi procesat
unset($doc1);
```

Funcţiile pe care le include bibloteca *XML22* se pot împărţi în două clase: funcţii care returnează tablouri de *tag*-uri (fragmente de document) şi funcţii care returnează un singur *tag*. Acelea care returnează o colecţie de *tag*-uri realizează acest lucru prin intermediul unor tablouri indexate multidimensionale şi anume prin tablouri de tip *tag*, prezentate mai sus.

- funcţiile care returnează o colecţie de *tag*-uri sunt: `xml22_get_by_regex()`, `xml22_get_all_descendants()`, `xml22_get_all_ancestors()`, `xml22_get_all_siblings()`, `xml22_get_all_of_name()`, `xml22_get_all_of_content()`, `xml22_copy_fragment()`
- funcţiile care returnează un singur *tag* sunt: `xml22_create_tag()`, `xml22_add_attribute()`, `xml22_delete_attribute()`, `xml22_get_root()`, `xml22_get_by_id()`, `xml22_get_first_child()`, `xml22_get_last_child()`, `xml22_get_next_sibling()`, `xml22_get_prev_sibling()`

ANEXA E

Biblioteca *JpGraph*

Iavi Rotberg, pit@infoiasi.ro

Pentru a genera grafice se poate folosi biblioteca *JpGraph*, disponibilă la adresa **http://www.aditus.nu/jpgraph/**. *JpGraph* este o bibliotecă scrisă în PHP, în manieră orientată-obiect, care facilitează generarea unei multitudini de tipuri de grafice (*charts*), de la simple până la complexe. Biblioteca *JpGraph* este distribuită sub o licenţă duală QPL 1.0 (Qt-License) pentru uz necomercial şi JpGraph Professional License pentru aplicaţii comerciale. Pentru a folosi această bibliotecă este necesară o versiune PHP mai mare decât 4.0.2, compilată cu suport pentru biblioteca GD versiunea 2.0.1 (a se vedea şi cele descrise în capitolele 4 şi 6 ale cărţii de faţă).

Principalele facilităţi oferite sunt:

- scale flexibile, cu suport pentru scale numerice, text şi mixte;
- scale configurabile de către utilizator (de exemplu, scale diferite pentru coordonatele carteziene *x* şi *y*);
- scalare manuală şi automată;
- poziţii configurabile ale axelor;
- suport pentru principalele formate grafice folosite pe Web: PNG, GIF şi JPEG;
- suport pentru fonturi de tip *True Type*;
- mecanism de *cache* pentru a nu supraîncărca serverul HTTP;
- legende pentru grafice configurabile automat şi manual;
- design modular, asigură doar includerea codului necesar;
- suport pentru *anti-aliasing*, *alpha-blending*, degradeuri (*gradient fill*), imagini de fundal, 400 culori.

Drept o regulă generală putem considera faptul că fiecare script PHP care generează un grafic trebuie să fie inclus într-un fişier separat, care va fi referenţiat în pagina Web prin marcatorul `<img>`. De exemplu, codul XHTML de mai jos include în cadrul paginii graficul generat de scriptul PHP `grafic.php`:

```
<img src="grafic.php" alt="Dinamica vanzarilor" />
```

Biblioteca va genera automat anteturile HTTP necesare (de exemplu, *Content-Type*), care vor fi transmise navigatorului pentru a recunoaşte corect formatul imaginii.

Pentru a alege formatul imaginii graficului există două posibilităţi: schimbăm în fişierul de configurare pe care biblioteca îl citeşte în mod automat valoarea „auto" cu una dintre valorile „PNG", „JPG" sau „GIF":

```
DEFINE ("DEFAULT_GFORMAT" ,"auto");
```

sau se poate seta în cadrul scriptului, cu ajutorul funcţiei *SetImgFormat()*:

```
$graph->img->SetImgFormat("jpeg");
```

Biblioteca *JpGraph* suportă fonturi de tip *True Type*, dar pentru a face uz de acestea trebuie definit un director care va conţine fonturile dorite. Acest lucru se poate realiza modificând următoarea linie din fişierul de configurare:

```
DEFINE ("TTF_DIR","/usr/local/fonts/ttf/");
```

Este preferabil ca fonturile *True Type* să fie folosite numai pentru titluri, subtitluri, legende, deoarece acestea se procesează implicit *anti-aliasing* ceea ce conduce la un timp mai îndelungat de prelucrare.

Pentru a nu supraîncărca serverul HTTP se poate activa mecanismul de *cache*. Pentru aceasta, trebuie modificate liniile de mai jos din fişierul de configurare:

```
DEFINE("USE_CACHE", true);
DEFINE("READ_CACHE", true);
```

iar apoi trebuie definit un director pentru a menţine informaţiile dorite:

```
DEFINE ("CACHE_DIR", "/tmp/jpgraph_cache/");
```

Pentru a folosi efectiv mecanismul de *cache* trebuie specificat un nume de fişier care să menţină imaginea după generarea ei. Dacă nu este specificat un anumit fişier, se va aloca unul în mod automat, care va avea acelaşi nume cu scriptul. De asemenea, se poate specifica şi o perioadă de timp după care fişierul va fi şters din *cache*. De exemplu, următoarea linie creează un fişier numit în mod automat cu numele scriptului şi care va fi păstrat în *cache* timp de 60 de minute:

```
$graph = new Graph(300, 200, "auto", 60);
```

Proprietăţile – datele membre – ale clasei *Graph* oferite de biblioteca *JpGraph* (comune tuturor tipurilor de grafice) sunt:

- *Graph::title*, *Graph::subtitle* şi *Graph::subsubtitle* pentru a seta un titlu, un subtitlu şi respectiv un subsubtitlu.
- *Graph::legend* pentru a stabili o legendă.
- *Graph::footer::left*, *Graph::footer::center* şi *Graph::footer::right* pentru a seta un picior-de-pagină (*footer*) în partea stânga, centru, respectiv dreaptă.
- *Graph::xaxis*, *Graph::yaxis* pentru a seta coordonatele *x* şi *y*;
- *Graph::xgrid*, *Graph::ygrid* pentru a seta axele de tip rastru *x* şi *y*.

Metodele principale ale clasei *Graph* (comune tuturor tipurilor de grafice) sunt următoarele:

- *Graph::Add()* adaugă unui grafic obiectele care vor fi desenate: linii, bare, text etc.
- *Graph::SetMargin()* setează o margine pentru titluri, subtitluri etc.
- *Graph::SetColor()* setează o culoare.
- *Graph::SetShadow()* creează efectul de umbrire a graficului.
- *Graph::SetGridDepth()* setează adâncimea suprafeţei efective a graficului.

- *Graph::SetAngle()* setează un unghi de rotire a axei.
- *Graph::SetBackgroundImage* setează o imagine de fundal.

Tipurile de grafice care pot fi generate sunt: liniar (*line-plots*), liniar umplut (*filled line-plots*), liniar de acumulare (*accumulated line-plots*), bare (*bar plots*), bare de acumulare (*accumulated bar plots*), bare grupate (*grouped bar plots*), erori (*error plots*), dispersie (*scatter plots*), radar (*radar plots*), plăcintă 2D şi 3D (*2D and 3D pie charts*).

ANEXA F

Introducere în WML

Marius Ropotă, ropo@infoiasi.ro
Sabin Buraga, busaco@infoiasi.ro

Limbajul *WML* (*Wireless Markup Language*) este un limbaj de marcare, bazat pe XML, pentru redarea informaţiilor destinate a fi reprezentate pe ecranele dispozitivelor mobile. Limbajul WML poate fi folosit deci pentru a crea pagini destinate a fi afişate într-un *browser WAP* (*Wireless Application Protocol*). Actualmente, există mai multe versiuni operaţionale precum WAP 1.1, WAP 1.2, WAP 1.3 sau WAP 2.0.

Aparatele de genul telefoanelor mobile sunt caracterizate prin posibilităţi reduse de afişare a datelor. Prin urmare, paginile destinate a fi redate pe asemenea aparate vor trebui să respecte acest considerent şi, de asemenea, să ofere o portabilitate crescută. Din ambele cauze, HTML nu constituie alegerea potrivită, pentru aceasta adoptându-se limbajul WML, dezvoltat de comunitatea *WAP Forum*, formată din diverse companii interesate în domeniu.

Primul limbaj dedicat aparaturii mobile a fost *HDML* (*Handheld Device Markup Language*), surclasat ulterior de dezvoltarea continuă a limbajului WML.

Fişierele WML prezintă două forme: ca fişiere sursă de tip text, incluzând marcatori specifici şi stocate cu extensia `.wml` şi drept fişiere compilate (`.wmlc`), conţinând o formă compactă şi independentă de platformă a fişierului sursă original.

Fiecare fişier WML debutează cu o declaraţie a adresei unui fişier DTD de validare, ceea ce permite *browser*-elor care „înţeleg" XML (inclusiv cele încorporate în aparatele mobile) să verifice corectitudinea codului WML.

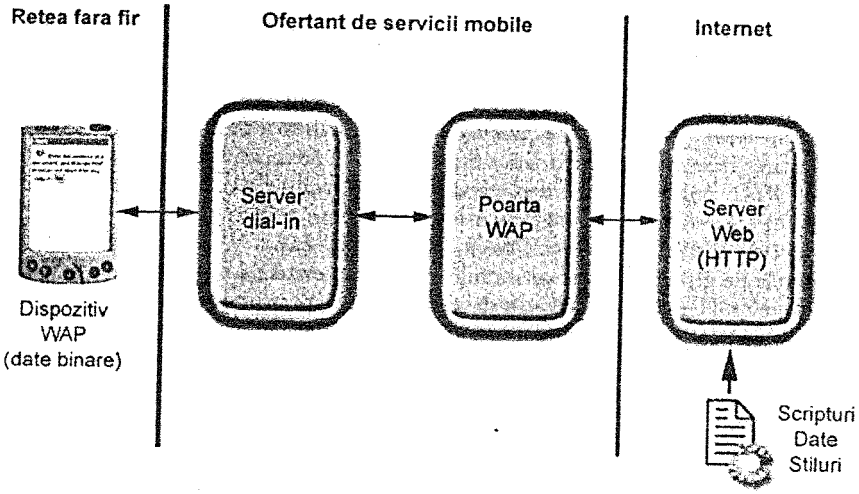
Pentru a manipula unele construcţii WML în mod dinamic s-a creat un limbaj de tip *script* specific, denumit *WMLScript*, inspirat de JavaScript. Codul-sursă poate fi conţinut fie în fişiere text-sursă (`.wmls`), fie în fişiere compilate (`.wmlsc`).

Posibilităţile grafice ale dispozitivelor mobile fiind reduse, a fost creat un format grafic special *WBMP* (*Wireless Bitmap*). Acest format este unul de tip rastru (*bitmap*). Deja există o serie de instrumente de creare a fişierelor WBMP, dintre care se pot menţiona *WAP Pictus*, *WAP Draw* ori *Adobe Photoshop*.

Comunicarea dintre clienţii WAP şi servere se realizează prin intermediul unui protocol special. Dispozitivele mobile dispun de mult mai puţine resurse (de exemplu, putere a procesorului, dimensiune a memoriei şi a ecranului de afişare) decât calculatoarele *desktop* actuale. Pentru a răspunde necesităţilor comunicaţiilor mobile s-au dezvoltat o serie de tehnologii, dintre care se disting protocoalele *WAP* (*Wireless Application Protocol*) şi *I-mode* şi diverse medii de dezvoltare precum *Java 2 Micro Edition (J2ME)*.

Protocolul WAP reprezintă în fapt un set de specificaţii de tip *open-source* care adaptează tehnologiile Internet existente (de exemplu, suita de protocoale TCP/IP) şi introduce câteva concepte specifice comunicaţiilor cu dispozitivele mobile.

Standardul WAP pune la dispoziţia dezvoltatorilor două elemente de bază: un protocol la nivel de aplicaţie şi un mediu de lucru bazat pe un navigator WAP. Protocolul este orientat client/server (astfel, clienţii WAP vor avea nevoie de un server WAP oferind diverse servicii clienţilor). Partea de server este denumită şi *poartă* (*gateway*) WAP, având rolul de redirecţionare a cererilor clienţilor WAP către un server clasic (de exemplu, server Web sau FTP). Astfel, poarta poate fi considerată ca fiind un mediator între Internet şi un dispozitiv folosind WAP.



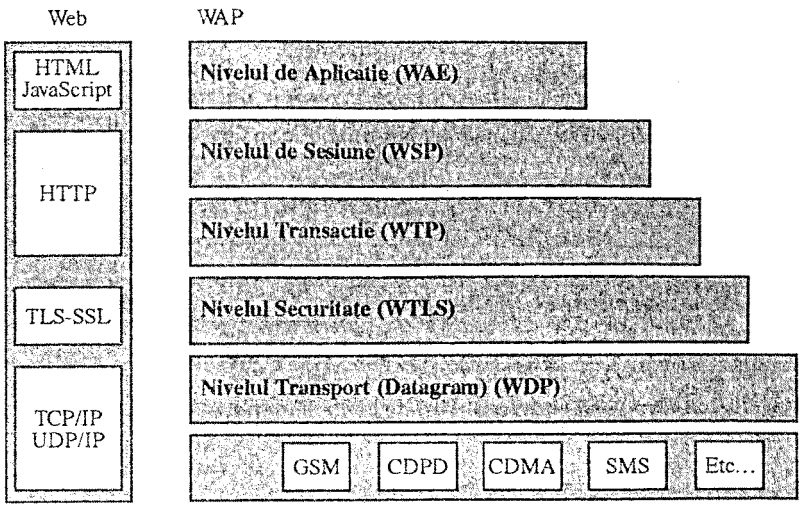
Comunicarea între dispozitive WAP şi Internet

Între reţeaua fără fir reprezentată de dispozitivele WAP şi Internet se interpune un ofertant de servicii (*wireless service provider*) care pune la dispoziţie diverse servicii de comunicare fără fir (ca exemple de astfel de ofertanţi se pot menţiona AT&T, British Telecom, Verizon ori Zapp).

Modelul de programare WAP are la bază modelul de programare Web adaptat însă caracteristicilor mediului fără fir. Rolul porţii WAP este de a converti paginile trimise de serverul Web într-un format ce poate fi afişat pe ecranele dispozitivelor mobile. Pentru a optimiza aplicaţiile se poate recurge şi la un server *proxy* (opţional în specificaţiile WAP 2.0).

WAP foloseşte o altă stivă de protocoale faţă de stiva clasică TCP/IP. În figura de mai jos sunt prezentate nivelurile WAP, acronimele reprezentând: *Wireless Application Environment*, *Wireless Session Protocol*, *Wireless Transaction Protocol*, *Wireless Transport Layer Security* şi *Wireless Datagram Protocol*.

Cel mai important nivel din punctul de vedere al unui dezvoltator Web este *WAE* (*WAP Application Environment*) care reprezintă de fapt nivelul la care se situează clientul WAP, adică navigatorul (*browser*-ul) procesând date marcate în WML.



Stiva de protocoale WAP

Structura unui document WML este simplă: acesta e structurat în pachete (*deck*) de cărţi de joc (*card*). Un pachet e desemnat de *tag*-urile `<wml>...</wml>` şi poate conţine diverse meta-informaţii (date prin marcatori `<meta>`) şi un număr de elemente `<card>`. Cărţile de joc sunt utilizate pentru navigarea între ecrane de informaţie. Se dă posibilitatea utilizatorului să selecteze un `<card>` făcând parte din alt `<deck>`.

Un scurt exemplu (pentru a stabili ce versiune de limbaj WML utilizăm, se foloseşte declarea tipului de document prin construcţia `<!DOCTYPE...>`):

```
<?xml version="1.0" ?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
          "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id="identificator" title="Titlu">
    <p>Acesta este <b>continutul</b>:</p>
    <p>Salutari din <i><u>WML</u></i>!</p>
  </card>

</wml>
```

După cum se poate observa, pentru formatarea informaţiilor se pot folosi unii marcatori prezenţi şi în HTML (vezi mai jos). Sunt permise şi ancorele (legăturile) spre alte documente, pentru aceasta recurgându-se la marcatorii `<a>` şi `<anchor>`. Pentru navigarea între diferite *card*-uri se pot utiliza elementele `<go>`, `<prev>` sau `<refresh>`. Interacţiunea cu utilizatorul se realizează prin intermediul unor construcţii similare celor de la HTML (se acceptă marcatori de genul `<select>`, `<option>` ori `<input>`). Anumite evenimente ce pot surveni (de exemplu, *timer*, *onpick*, *onenterforward* etc.) pot fi tratate

prin <do> sau <onevent>. Se pot folosi şi diferite variabile, pentru accesul la diverse informaţii globale sau la datele introduse de utilizatori.

Pentru ca serverul Web (Apache) să interpreteze corect tipurile de documente WML şi imaginile WBMP care pot fi vizualizate de clienţii WAP, trebuie specificate următoarele tipuri MIME (adăugate în fişierul de configuraţie `httpd.conf` ori într-un fişier `.htaccess`):

```
# tipuri MIME pentru WAP:
# * pentru cod PHP
AddType application/x-httpd-php .wml
# * pentru documente WML de sine-statatoare
AddType text/vnd.wap.wml .wml
# * pentru fisiere grafice folosite de WAP
AddType image/vnd.wap.wml .wbmp
```

Lista elementelor WML 1.1 uzual folosite este cea prezentată în tabelele următoare.

Elemente de structură

Element	Scop
<access>	Specifică informaţii despre accesul la un *deck*
<card>	Defineşte un *card* într-un *deck*
<head>	Conţine informaţii despre document
<meta>	Defineşte meta-informaţii despre document
<template>	Defineşte un şablon pentru toate *card*-urile dintr-un *deck*
<wml>	Defineşte un *deck* (element-rădăcină WML)
<!--... -->	Comentariu

Elemente pentru organizarea informaţiilor

Element	Scop
 	Linie nouă (*line break*)
<p>	Paragraf
<table>	Defineşte un tabel
<td>	Defineşte o celulă de tabel
<tr>	Defineşte un rând într-un tabel

Elemente de formatare a textului

Element	Scop
<b>	Text îngroşat (*bold*)
<big>	Text de dimensiuni mai mari
	Text intensificat
<i>	Text scris italic
<small>	Text de dimensiuni mai mici
	Text îngroşat
<u>	Text subliniat

Elemente pentru hiper-legături

Element	Scop
<a>	Specifică o ancoră (*link*)
<anchor>	Specifică o ancoră (*link*)

Elemente pentru inserare de imagini

Element	Scop
<img>	Inserează o imagine în document

Elemente pentru evenimente

Element	Scop
<do>	Declanşează o acţiune atunci când utilizatorul va acţiona cu un *click* pe conţinutul *tag*-ului
<onevent>	Conţine cod care va fi executat atunci când apare unul dintre evenimentele următoare: *onenterbackward, onenterforward, onpick, ontimer*
<postfield>	Conţine informaţii care vor fi trimise către server împreună cu *tag*-ul <go>

Elemente pentru acţiuni

Element	Scop
<go>	Specifică acţiunea de a schimba *card*-ul curent
<noop>	NOOP = NO Operation. Specifică faptul că nu se va executa nimic. Folosit pentru a rescrie *tag*-urile dintr-un *deck*
<prev>	Specifică acţiunea de a merge înapoi la *card*-ul precedent
<refresh>	Reîmprospătează variabilele specificate. Dacă variabilele sunt afişate pe ecran, se reîmprospătează şi ecranul

Elemente pentru interacţiunea cu utilizatorul

Element	Scop
<fieldset>	Folosit pentru a grupa elemente într-un *card*
<input>	Defineşte un câmp de intrare (*input-field*)
<optgroup>	Defineşte o zonă de selecţie
<option>	Defineşte o opţiune dintr-o zonă *select*
<select>	Defineşte o listă de selecţie (*select list*)

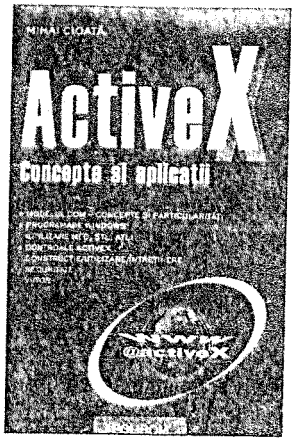
Elemente pentru lucrul cu variabile

Element	Scop
<setvar>	Atribuie unei variabile o valoare specificată. Se foloseşte împreună cu <go>, <prev> sau <refresh>
<timer>	Defineşte un cronometru (*timer*)

Bibliografie generală

1. D. Acostăchioaie, *Administrarea şi configurarea sistemelor Linux* (ediţia a doua), Polirom, Iaşi, 2003
2. D. Acostăchioaie, *Securitatea sistemelor Linux*, Polirom, Iaşi, 2003
3. A. Ames, D. Nadeau, J. Moreland, *VRML 2.0 Sourcebook*, John Wiley & Sons, 1997
4. T. Berners-Lee, *Weaving the Web*, Orion, 1999
5. T. Bidwell *et al.*, *Hack Proofing Your E-commerce Site*, Syngress Publishing, 2001
6. T. Butzon, *PHP by Example*, QUE, 2001
7. S. Buraga, *Tehnologii Web*, Matrix Rom, Bucureşti, 2001: http://www.infoiasi.ro/~busaco/books/web.html
8. S. Buraga, *Proiectarea siturilor Web*, Polirom, Iaşi, 2002: http://www.infoiasi.ro/~design/
9. S. Buraga, V. Tarhon-Onu, Ş. Tanasă, *Programare Web în bash şi Perl*, Polirom, Iaşi, 2002: http://www.infoiasi.ro/~cgi/
10. S. Buraga, G. Ciobanu, *Atelier de programare în reţele de calculatoare*, Polirom, Iaşi, 2001: http://www.infoiasi.ro/~lrc/
11. K. Egervari, J. Parise, J. Coggershall, *Professional PHP 4 Programming*, Wrox Press, 2002
12. M. Fotache, *SQL*, Polirom, Iaşi, 2001
13. I. Gilfillan, *Mastering MySQL 4*, Sybex, 2003
14. J. Greenspan, B. Bulger, *MySQL/PHP Database Applications*, M&T Books, 2001
15. B. Laurie, P. Laurie, *Apache: The Definitive Guide* (2nd Edition), O'Reilly & Associates, 1999
16. R. Lerdorf, K. Tatroe, Bob Kaehms, R. McGredy, *Programming PHP*, O'Reilly & Associates, 2002
17. Bill McCarty, *PHP 4*, Teora, Bucureşti, 2002
18. J. Meloni, *PHP Fast & Easy Web Development* (2nd Edition), Premier Press, 2002
19. T. Ratschiller, T. Gerken, *Web Application Development with PHP 4.0*, New Riders Publishing, 2000
20. N. Shi, V.K. Murthy, *Architectural Issues of Web-Enabled Electronic Business*, Idea Group Publishing, 2003
21. S. Suehring, *MySQL Bible*, Wiley Publishing, 2002
22. Ş. Tanasă, C. Olaru, Ş. Andrei, *Java de la 0 la expert*, Polirom, Iaşi, 2003
23. Ş. Trăuşan-Matu, *Interfaţarea evoluată om-calculator*, Matrix Rom, Bucureşti, 2000
24. Ş. Trăuşan-Matu *et al.*, *Prelucrarea documentelor folosind XML şi Perl*, Matrix Rom, Bucureşti, 2001
25. H.E. Williams, D. Lane, *Web Database Applications with PHP & MySQL*, O'Reilly & Associates, 2002
26. L. Wroblewski, *Site-Seeing – A Visual Approach to Web Usability*, Hungry Minds Inc., 2002
27. * * *, *Apache*: http://www.apache.org/
28. * * *, *Apache Week*: http://www.apacheweek.com/
29. * * *, *Code Walkers*: http://codewalkers.com/

30. * * *, *Developmentor*: http://www.develop.com/tutorials/
31. * * *, *Dev Shed – PHP*: http://www.devshed.com/Server_Side/PHP
32. * * *, *Digital Web*: http://www.digital-web.com/
33. * * *, *EL ROUBIO – Le PHP des éléPHPants et des PHPin'ups*: http://www.elroubio.net/
34. * * *, *IBM Developer's Site*: http://www.ibm.com/developer/
35. * * *, *Lighthouse 3D*: www.lighthouse3d.com
36. * * *, *Microsoft Developers Network*: http://msdn.microsoft.com/
37. * * *, *Oasis*: http://www.oasis-open.org/
38. * * *, *PHP*: http://www.php.net/
39. * * *, *PHP Beginner*: http://www.phpbeginner.com/
40. * * *, *PHP Builder*: http://www.phpbuilder.com/
41. * * *, *PHP Classes*: http://www.classes.net/
42. * * *, *PHP Freaks*: http://www.phpfreaks.com/
43. * * *, *PHP Romania*: http://www.phpromania.as.ro/
44. * * *, *PHP Talk*: http://www.phptalk.com/
45. * * *, *PHP World*: http://www.phpworld.com/
46. * * *, *Rapoartele tehnice ale Consorțiului Web*, Boston, 2003: http://www.w3.org/TR/
47. * * *, *Romanian Web Developers*: http://rowd.zuavra.net/
48. * * *, *RoPHP*: http://www.rophp.net/
49. * * *, *SourceForge*: http://www.sourceforge.net/
50. * * *, *SVG Zone*, Adobe: http://www.adobe.com/svg/
51. * * *, *VisiBooks*: http://www.visibooks.com/
52. * * *, *W3Schools*: http://www.w3schools.com/
53. * * *, *WAP Forum*: http://www.wapforum.org/
54. * * *, *WAP Links*: http://www.waplinks.com/
55. * * *, *Web 3D*: http://www.w3d.org/
56. * * *, *Web Builder*: http://webbuilder.netscape.com/
57. * * *, *WebLogs*: http://www.weblogs.com/
58. * * *, *Web Services*: http://www.webservices.org/
59. * * *, *XML.org*: http://www.xml.org/
60. * * *, *XML-RPC*: http://www.xmlrpc.com/
61. * * *, *XML Software*: http://www.xmlsoftware.com/
62. * * *, *XML Standards Library*: http://www.mobileworkspace.com/xmlstds/
63. * * *, *XML Times*: http://www.xmltimes.com/
64. * * *, *Zend – the PHP Company*: http://www.zend.com/
65. * * *, *ZVON*: http://www.zvon.org/

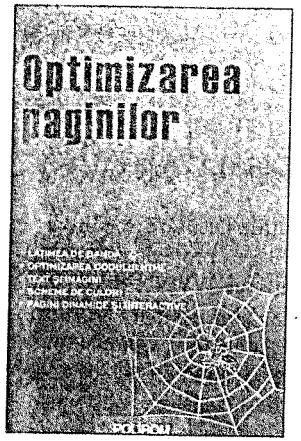


Mihai Cioată

ACTIVE X. CONCEPTE ŞI APLICAŢII

Cartea se adresează programatorilor de nivel mediu, care vor să înțeleagă această tehnologie Microsoft pentru a putea merge mai departe în conceperea de aplicații destinate sistemului de operare Windows și pentru a dezvolta aplicații bazate pe modelul COM, în general, și controale Active X, în particular. Lucrarea are o construcție graduală, pornindu-se de la concepte și termeni-cheie, utili înțelegerii modelului de programare COM, și continuând cu tehnici de construire a controalelor Active X, în medii precum Delphi și Visual C++, medii care asigură succesul în ceea ce privește viteza cu care pot fi create respectivele componente. De asemenea, se prezintă modul de construcție de controale Active X performante prin utilizarea șabloanelor ATL oferite de mediul Visual Studio. Cititorii pot afla amănunte privitoare la unele aspecte interesante ale programării în C++ (șabloane, biblioteci de șabloane standard – STL, „pointeri inteligenți") și ale programării de sistem Windows. În plus, se dau detalii referitoare la utilizarea controalelor Active X oferite de navigatorul Internet Explorer direct din cadrul paginilor Web (de exemplu, manipularea fișierelor sau procesarea documentelor XML). Studiile de caz care încheie lucrarea prezintă diferite „rețete" de creare a controalelor Active X în Visual Basic și Visual C++ și de utilizare a acestora în diverse contexte (de exemplu, pe Web).

Modelul COM – concepte și particularități • Programare Windows (utilizare MFC, STL, ATL) • Controale Active X. Construcție/ Utilizare/ Întreținere/ Securitate/ Viitor



Calin Ioan Acu

OPTIMIZAREA PAGINILOR WEB

O pagină Web trebuie să stârnească interesul. Conținutul ei nu se poate reduce la câteva fraze scurte, dar nici nu poate fi un amalgam de sunete și imagini. Situl trebuie să comunice mult, cu mijloace puține. Trebuie să fie afișat imediat, dar să fie spectaculos și atractiv. Cum să navigăm rapid și fără sincope? Ce modificări trebuie aduse sitului pentru a-l optimiza și a-l face mai ușor de accesat? Iată doar o parte din întrebările la care volumul încearcă să ofere un răspuns, structurat astfel încât să fie accesibil și începătorilor (pentru care există explicații detaliate, exemple practice și un cuprinzător studiu de caz), și celor avansați, care doresc să-și verifice și valideze soluțiile folosite.

Lățimea de bandă • Optimizarea codului HTML • Utilizarea și reutilizarea imaginilor • Text sau imagini? • Divizarea imaginilor • Utilizarea thumbnails-urilor • Intercalarea și transparența

Sabin Buraga, Victor Tarhon-Onu, Ştefan Tanasă

PROGRAMARE WEB ÎN BASH ŞI PERL

Lucrarea descrie în detaliu unul dintre cele mai folosite standarde de programare pe partea server a aplicaţiilor destinate Web-ului: CGI (*Common Gateway Interface*), punând un accent deosebit asupra elaborării de aplicaţii CGI în două dintre limbajele de tip *script*, larg utilizate pe platforma Linux: *shell*-ul bash şi limbajul Perl.
Autorii insistă asupra prelucrării practice a datelor prin intermediul serverelor de baze de date relaţionale (*e.g.* MySQL sau PostgreSQL) şi a documentelor XML (*Extensible Markup Language*).
Numeroasele studii de caz prezentate demonstrează flexibilitatea şi uşurinţa în realizarea de aplicaţii Web complexe, utilizând exclusiv tehnologii şi sisteme deschise.
CD-ul care însoţeşte lucrarea cuprinde atât codurile-sursă ale majorităţii exemplelor şi studiilor de caz prezentate, cât şi documentaţii suplimentare, programe şi aplicaţii utile pentru experimentarea şi aprofundarea materialului din carte, disponibile pentru diferite distribuţii Linux sau alte platforme.

• Protocolul HTTP (*HyperText Transfer Protocol*) • Scheme XML • Limbajele de marcare şi aplicaţiile lor • Standardul CGI (*Common Gateway Interface*) • Cookie-urile • Programarea CGI în bash • Scripturi CGI în bash • Limbajul şi modulele Perl • Programarea CGI în Perl • Perl şi bazele de date relaţionale • Prelucrarea documentelor XML • Migrarea de la baze de date relaţionale la documente XML

Sabin Buraga

PROIECTAREA SITURILOR WEB. DESIGN ŞI FUNCŢIONALITATE

Cartea conţine toate informaţiile necesare, explicate pe înţelesul tuturor, pentru realizarea profesionistă a paginilor Web: arhitectura spaţiului WWW; limbaje de marcare: HTML, XHTML, XML; tipuri de situri; structura unui site; soluţii de proiectare a conţinutului; link-uri şi conexiuni interne; strategii de căutare; realitatea virtuală, modele de afaceri pe Web; realizarea unui magazin virtual; mijloace de plată electronică. CD-ul care însoţeşte cartea dă imaginea completă a construirii şi utilităţii siturilor Web: studii de caz; documentaţii; curs de tehnologii Web (14 lecţii); 100 de aplicaţii şi utilitare pentru realizarea şi întreţinerea siturilor; aplicaţii pentru comerţ electronic; colecţii de ilustraţii şi pictograme.

www.polirom.ro

Redactor: Andrei Dimitriu
Coperta: Ionuţ Broştianu
Tehnoredactor: Lucian Pavel

Bun de tipar: octombrie 2003. Apărut: 2003
Editura Polirom, B-dul Carol I nr. 4 • P.O. Box 266
700506, Iaşi, Tel. & Fax: (0232) 21.41.00; (0232) 21.41.11;
(0232) 21.74.40 (difuzare); E-mail: office@polirom.ro
Bucureşti, B-dul I.C. Brătianu nr. 6, et. 7, ap. 33,
O.P. 37 • P.O. Box 1-728, 030174
Tel.: (021) 313.89.78; E-mail: polirom@dnt.ro

Tiparul executat la S.C. LUMINA TIPO s.r.l.
str. Luigi Galvani nr. 20 bis, sect. 2, Bucureşti
Tel./Fax: 211.32.60, 212.29.27, E-mail: lumina-lex@fx.ro